IMS
15.1.0

Communications and Connections (2024-08-30 edition)

SC27-6783-00

Note

Before you use this information and the product it supports, read the information in “Notices” on page 953.

2024-08-30 edition.

This edition applies to IMS 15 (program number 5635-A06), IMS Database Value Unit Edition, V15.01.00 (program number 5655-DS5), IMS Transaction Manager Value Unit Edition, V15.01.00 (program number 5655-TM4), and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 1974, 2020.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this information..... xxiii
Prerequisite knowledge..... xxiii
How new and changed information is identified..... xxiii
How to read syntax diagrams..... xxiii
Accessibility features for IMS 15..... xxv
How to send your comments..... xxv

Part 1. Configuring external Java environment connections..... 1

Chapter 1. IMS Universal drivers: configuring connections to IMS..... 3
Configuring WebSphere Application Server for EJB development with the IMS Universal drivers..... 3
IMS Universal drivers: WebSphere Application Server type-4 connections..... 4
Installing a type-4 IMS Universal Database resource adapter on WebSphere Application Server..... 5
Defining a connection factory for a type-4 IMS Universal Database resource adapter on WebSphere Application Server..... 6
Installing an EAR file that uses a type-4 IMS Universal Database resource adapter on WebSphere Application Server..... 7
IMS Universal drivers: WebSphere Application Server for z/OS type-2 connections..... 7
Installing a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS..... 8
Optional: set the WebSphere Application Server for z/OS classpath for applications that use a type-2 IMS Universal Database resource adapter..... 9
Defining a connection factory for a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS..... 10
Installing an EAR file that uses a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS..... 11
IMS Universal drivers: WebSphere Application Server Liberty type-4 connections..... 12
WebSphere Application Server Liberty type-4 connections sample server.xml configuration file..... 14
IMS Universal drivers: WebSphere Application Server Liberty type-2 connections..... 15
WebSphere Application Server Liberty type-2 connections sample server.xml configuration file..... 18
The IMS Universal drivers: CICS connections..... 19
Configuring CICS for the type-2 IMS Universal drivers..... 20
Running applications on CICS that use the type-2 IMS Universal drivers..... 21

Part 2. CPI Communications and APPC/IMS..... 23

Chapter 2. CPI Communications..... 25
CPI-C driven application programs..... 25
SAA resource recovery commit processing..... 25
Normal termination..... 25
Backout processing..... 25
Abnormal termination..... 26
Session failure..... 26
Return codes..... 26
System restart/resolve-in-doubt processing..... 26
CPI-C application program recovery..... 27
Programming requirements..... 27

Pseudonym files....27
RRS and distributed syncpoint/protected conversations....27
The two-phase commit protocol....28
Local-resource recovery versus distributed-resource recovery....29
IMS as a resource manager....30
Activating protected conversations....30

Chapter 3. Administering APPC/IMS and LU 6.2 devices....33
APPC/IMS overview....33
APPC/IMS flood control....34
APPC/IMS application program interface....35
APPC/IMS application programs....35
Standard IMS application programs....36
MSC and standard IMS application programs....36
Modified IMS application programs....37
MSC and modified IMS application programs....38
CPI Communications driven application programs....38
Using the MOD name and LTERM interface....39
Establishing APPC/IMS....40
TP_Profile....40
APPC/MVS Administration utility (ATBSDFMU) example....41
Outbound LU specification....42
Outbound side information....43
PARMLIB member....43
APPC/MVS Timeout Service....44
APPC/MVS Error Extract Service....45
Initializing and changing LU 6.2 descriptors....45
Using MSC in an APPC/IMS environment....46
Recovering APPC transactions in an MSC environment....47
Recoverable versus nonrecoverable transactions....47
Local APPC transaction discardability versus nondiscardability....47
Transaction processing point of failure....48
Recoverability flows of LU 6.2 transactions....50
Planning for XRF and APPC....52
Transaction retry characteristics....52
Qualifying network LU names....52
Managing multiple LUs for a single IMS system....53
Reassigning an LU to another IMS system....53
DFSAPPC system service....54
Message switching....54
Asynchronous output delivery....56
APPC transaction security....57

Part 3. Extended Terminal Option (ETO)....59

Chapter 4. Overview of the Extended Terminal Option....61
ETO terminology....61
ETO descriptors....63
ETO concepts....65
Descriptors and exit routines....65
How descriptors are created and used....66
Summary of ETO implementation....66

Chapter 5. Administering the Extended Terminal Option....69
Planning for ETO....69
Identifying your requirements....70
ETO restrictions....70

Defining physical terminals ... 70
Identifying VTAM device types, screen sizes, and models ... 74
Planning a high-security environment with ETO ... 78
Planning for MFS ... 79
Planning user IDs ... 79
Planning user queue names ... 79
Planning operations ... 80
Planning for MSC support with ETO ... 80
Coding ETO descriptors ... 80
Creating descriptors using the system definition process ... 81
Storing descriptors ... 81
Creating logon descriptors ... 81
Creating user descriptors ... 84
Creating MFS device descriptors ... 86
Creating MSC descriptors ... 88
Exit routines ... 88
Starting ETO ... 89
Logging onto ETO terminals ... 89
Limiting dynamic logon to specific terminal types ... 90
Creating and reusing LTERM control blocks ... 90
Using default CINIT or BIND user data formats ... 90
Signing on and queue LTERM allocation ... 91
Providing signon data ... 91
Providing signon data for ISC, SLU-P, Finance, and output-only devices ... 91
Signing on multiple times ... 91
Receiving DFS3649A, the signon required message ... 92
Receiving DFS3650I, the session status message ... 93
ETO terminal-LTERM relationship ... 93
How IMS determines which queues to allocate ... 94
Setting special processing modes ... 94
Printers with ETO ... 95
Direct printing ... 95
Associated printing ... 95
Defining your printers ... 96
Sharing printers using ETO ... 97
Operator commands ... 97
System definition parameters for ETO ... 98
Setting DEADQ status time with the DLQT parameter ... 98
Autosignoff (ASOT) ... 99
Autologoff (ALOT) ... 99
Autosignoff and autologoff timer ... 101
Autologon ... 101
Assigning output ... 101
Asynchronous output ... 101
Delivering output messages to non-originating terminals ... 102
Inadvertent output data streams ... 103
Signing off ... 103
Logging off ... 103
Improving performance by deleting ETO control blocks ... 103
IDC0 Trace facility ... 104
ETO and LU 6.1 (ISC) terminals ... 104
ETO and STSN terminals ... 105
SNA STSN terminal considerations ... 105
ETO and 3600/Finance and SLU P ... 105
/SIGN support for ETO STSN devices: ISC, Finance, and SLU P ... 106
Conversation mode and response mode with ETO ... 106

Part 4. External subsystem attach facilities..... 109

Chapter 6. DB2 Attach Facility 111
Preparing your system to use the DB2 Attach Facility..... 111
Managing how your Java dependent regions access Db2 for z/OS..... 111

Chapter 7. External Subsystem Attach Facility (ESAF)..... 113
What the external subsystem must provide..... 114
How external subsystems are specified to IMS..... 114
The basics of attach processing..... 115
Subsystem connections..... 116
Application call processing..... 118
Resource coordination..... 118
External subsystem command support..... 119
IMS services available to the ESAP..... 119

Chapter 8. Creating the external subsystem module table..... 121
DFSEMODL macro..... 121
DFSEWAL macro..... 124

Chapter 9. IMS External Subsystem Attach Facility processing..... 129
Loading the External Subsystem Attachment Package..... 129
Creating the EEVT control block..... 129
Loading external subsystem modules..... 130
Creating work areas for the ESAP..... 131
Initiating the external subsystem connection..... 131
Deferring the control region identify..... 132
Using the IMS Subsystem Startup Service..... 132
Establishing dependent region connections..... 132
Notify message..... 133
Application program request support..... 134
Language interface definition..... 134
Language interface entry points unique to external subsystems..... 134
Accessing multiple external subsystems..... 135
Resource recovery token..... 135
Terminating the external subsystem connection..... 136
Termination requested by the external subsystem..... 136
Dependent region connections..... 137
Explanation of stopped status..... 137

Part 5. IMS Connect and TCP/IP communications..... 139

Chapter 10. Overview of IMS Connect..... 141
IMS Connect client support..... 142
IMS Connect support for access to IMS DB..... 145
IMS Connect support for the IMS TM Resource Adapter..... 146
IMS Connect support for command requests to Operations Manager (OM)..... 147
IMS Connect support for ISC TCP/IP communications..... 148
IMS Connect support for IMS-to-IMS TCP/IP communications..... 150
MSC and IMS-to-IMS TCP/IP communications..... 151
OTMA and IMS-to-IMS TCP/IP communications..... 153
Overview of IMS Connect XML Conversion Support..... 154
IMS Connect support for z/OS Sysplex Distributor..... 155
Overview of IMS Connect security..... 156
Overview of defining and invoking IMS Connect..... 157

Chapter 11. Overview of IMS Connect exit routines....159
Overview of user message exit routines....160
Security and the IMS Connect user message exit routines....161
User-defined messages....161
Overview of function-specific exit routines....162
Macros that support IMS Connect exit routines....162
Exit interface blocks....163
XIB exit interface block for connections to IMS TM....163
XIBDS exit interface block for IMS TM data store information....164
XIB1 exit interface block for connections to IMS DB....167
XIBOD exit interface block for ODBM and IMS DB data store information....168

Chapter 12. IMS Connect support for IMSplex and shared queues....173
IMS Connect support for IMSplex....173
IMSplex support environment....173
Installing IMS Connect support for IMSplex....174
Retrieving ALTPCB output in a shared queues environment....174

Chapter 13. IMS Connect security support....175
IMS Connect support for RACF....175
Enabling generic return codes or message for RACF verifications....176
Enabling RACF security checking in IMS Connect....176
Enabling RACF security statistics for IMS Connect....177
IMS Connect default RACF user ID....178
IMS Connect RACF user ID cache....178
IMS Connect security for clients of IMS DB....179
Passing network security credentials through IMS Connect....179
Securing IMS-to-IMS TCP/IP connections....181
IMS Connect security exit routine....183
IMS Connect security and the OTMARTUX user exit....183
HWSSMPL0 and HWSSMPL1 security actions....183
IMS Connect responses to errors on RACROUTE calls from the sample exits....185
IMS Connect password management....190
Changing RACF passwords by using client messages....190
Changing RACF password phrases by using client messages....191
Enabling mixed-case password support....192
IMS Connect support for RACF PassTicket....193
SSL and IMS Connect....198
Trusted-user support for IMS Connect messages....198
Specifying an OTMA ACEE aging value in the IMS Connect configuration member....199

Chapter 14. IMS Connect support for callout requests....201
Configuring user-written IMS Connect clients for synchronous callout requests....201
Format of synchronous callout messages....203
Retrieving synchronous callout requests with RESUME TPIPE....204
Acknowledging receipt of synchronous callout messages....205
Returning callout responses to IMS....207
Returning an error response to IMS....208

Chapter 15. IMS Connect XML message conversion....209
IMS Connect XML converters....210
Structure of the XML message....210
Message conversion example....212

Chapter 16. IMS Connect message structures....213
IRM structures for IMS Connect client messages....213
Format of fixed portion of IRM in messages sent to IMS Connect....214

Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines....218
Format of IRM extensions....227
Output structure from client exit....229
Other IMS Connect structures....229
Message structures and IMS Connect user message exit routines....231
Input messages from client....231
Output message to client....232
Message structures....233
Examples of message structures in a simple interaction....248

Chapter 17. OTMA header fields used by IMS Connect....251
OTMA message-control fields used by IMS Connect....251
OTMA state data fields used by IMS Connect....257
OTMA security data fields used by IMS Connect....270
OTMA user data fields used by IMS Connect....274
Notes to OTMA header tables....281

Chapter 18. IMS Connect protocols....283
Transaction restrictions and limitations....283
Commit mode and synchronization level definitions....283
IMS Connect protocol level....284
IMS Connect conversational support....285
OTMA conversational protocols....285
IMS Connect conversational protocols....287
Purging undeliverable commit-then-send output....291
Specifying the purge function for undeliverable commit-then-send output....292
When IMS purges undeliverable commit-then-send output....292
Rerouting commit-then-send output....293
Specifying the reroute function for commit-then-send output....293
Specifying a destination for rerouted output....294
When IMS reroutes commit-then-send output....294
Recoverable IMS transactions....295
Send-only protocol....296
Send-only with acknowledgment protocol....297
Send-only with error protocol....298
Send-only protocol with serial delivery protocol....298
Send-only protocol for synchronous callout responses....299
Socket connections....300
Persistent sockets....300
Transaction sockets....301
Non-persistent sockets....301
Setting socket types for IMS TM clients....301
Socket connections for IMS-to-IMS TCP/IP communications....302
Socket processing for transactions....303
Managing the number of sockets....304
Resolving duplicate client IDs....307
IMS Connect override for the z/OS TCP/IP KeepAlive interval....309
TCP/IP failures....309
IMS Connect timeout specifications....310
Timeout specifications for IMS DB clients....310
Timeout specifications for IMS TM clients....310
Timeout specifications for IMS-to-IMS connections....320
IMS Connect transaction expiration support....321
Setting a transaction expiration time with IMS Connect....322
Retrieval of output on OTMA tpipe hold queues....322
RESUME TPIPE/receive protocol....323
Implementing asynchronous output support....326

Retrieving output with parallel **RESUME TPIPE** requests ... 327
Managing the retrieval of output messages ... 330
Retrieving output from alternate OTMA tpipe hold queues ... 335
Defining groups for shared asynchronous output ... 336
Asynchronous output message flow ... 337
IMS Connect client call flows ... 338
IMS Connect dead letter queue (HWS$DLQ) ... 342
Ping support for IMS Connect ... 342

Chapter 19. IMS Connect two-phase commit support ... 345
Overview of two-phase commit protocol ... 345
Distributed two-phase commit support ... 346
Support for the IMS Universal drivers ... 346
Global (XA) transactions with the IMS Universal drivers ... 346
One-phase commit global transactions with the IMS Universal drivers ... 349
Support for the IMS TM Resource Adapter ... 352
Global (XA) transactions with IMS TM Resource Adapter ... 352
One-phase commit global transactions with IMS TM Resource Adapter ... 356

Chapter 20. Unicode considerations for IMS Connect ... 359
Message translation ... 359

Chapter 21. TCP/IP settings for IMS Connect ... 361

Part 6. IMS VTAM network administration ... 363

Chapter 22. Introduction to the IMS Transaction Manager network ... 365
IMS TM network overview ... 365
IMS Transaction Manager services ... 369
The Data Communication Control (DCCTL) environment ... 371
Operating an IMS network ... 372
The shared-queues environment ... 373
Benefits of using shared queues ... 374
Required components of a shared-queues environment ... 374
Overview of the Common Queue Server ... 376
Balancing sessions with generic resources ... 377
IMSplex terminal management ... 378
Benefits of managing resources with a resource structure ... 378
Shared TM resources ... 379
Resource name uniqueness ... 379
Resource type consistency ... 380
Fast Path expedited message handler ... 380

Chapter 23. Planning the network ... 383
Planning for network administration ... 383
Documenting network and terminal requirements ... 383
IMS terminal network ... 384
Terminal connections to IMS ... 384
XRF terminal support ... 385
Logical terminals (LTERMs) ... 385
APPC/IMS and LU 6.2 terminal support ... 385
IMS messages and their scheduling ... 386
Message flow within the IMS online system ... 388
Conversational transactions ... 389
Message switches ... 390
Designing logical terminal networks ... 390
Logical-terminal chains ... 391

Logical-terminal queues....392
Separating input and output devices....392
Logical and physical terminal relationships....393
Master terminal....393
Master terminals in an XRF complex....393
NTO terminals....394
Resource modes and states....394
Terminal and user operating modes....394
Terminal and user states....396
Resource status recovery....397
Planning for security....401
Authorizing transactions in a TM network....402
Authorizing commands in a TM network....402
Transaction command security....403
Password security....403
Security for APPC/IMS....403
Security for ETO....404
Planning for Fast Path terminals....404
Planning for the Extended Recovery Facility (XRF)....405
Using MNPS or USERVAR in XRF complexes....406
Terminals in an XRF complex....406
Specifying terminal support....408
Planning for Rapid Network Reconnect (RNR)....409
Specifying levels of support....410
Persistent Session Tracking....410
IMS shutdown and RNR....411
Using RNR with XRF or VGR....411
Terminal reconnect protocols....411
Signon security....412

Chapter 24. Defining the network....415
Preparing for the operational network....415
Coordinating IMS definition and network definition....416
Using IMS as a host subsystem....416
Defining VTAM nodes....416
Estimating VTAM storage requirements....417
Determining VTAM buffer pool values....417
Determining the NCP buffer pool values....417
Determining static and dynamic terminal signon requirements....417
Checking requirements for LOGON MODE tables....418
Specifying initial VTAM configurations....419
Using SON/COS support in IMS....419
Starting an IMS network....420
Session initiation....420
IMS transaction types and transaction states....421
Determining transaction states....422
Defining VTAM for Rapid Network Reconnect (RNR)....423
Defining the level of persistent support....423
Defining the level of RNR support....423

Chapter 25. Editing and formatting IMS messages....425
Message Format Service....425
MFS components....426
Administering MFS....430
Advantages to using MFS....431
MFS control blocks....432
Creating MFS formats with SDF II....433
Basic edit....434

IMS editing for Intersystem Communication (ISC) 435
Transparency option..... 436
Unprotected screen option..... 436
Bypassing MFS editing..... 436
Locking and unlocking the terminal keyboard..... 436
IMS sensitivity to nongraphic message data..... 437
Output message segment editing..... 437
Editing of input message segments by MFS..... 437
Editing of input message segments by basic edit..... 437
Controlling output devices..... 438
Small buffer devices..... 438
Controlling output..... 439
Using a printer component..... 439
Spooled output control..... 439
Using printer components of the IBM 3270 Information Display System..... 440
Specifying candidate printers..... 440
Operational considerations..... 441
Sharing printers between systems..... 441

Part 7. Intersystem Communication (ISC)..... 443

Chapter 26. Overview of Intersystem Communication..... 445
Comparison of ISC and MSC..... 446
IMS facilities available to ISC..... 447
Sample system configurations..... 450
ISC support for TCP/IP..... 452
ISC between IMS and CICS..... 452
Terminal device-dependent data..... 453
Passing CICS data to IMS..... 453

Chapter 27. VTAM facilities used for ISC connections..... 455
VTAM commands and indicators..... 456
Using the VTAM application programming interface..... 457
Specifying logon modes when establishing a connection..... 457
Design considerations for secondary logical units..... 458

Chapter 28. IMS facilities affected by ISC..... 459
Editing messages..... 459
Issuing IMS commands from an ISC session..... 460
Effects on parallel sessions..... 460
Using IMS test mode for ISC VTAM sessions..... 460
IMS control block storage on ISC parallel sessions..... 461
Relationship of ISC and IMS execution modes..... 461
External specification of execution modes..... 461
Internal definition of execution mode..... 462
Resultant processing mode during ISC VTAM communications..... 462
LTERM users (subpools) and components..... 463

Chapter 29. Designing communications using the ISC protocol..... 467
Determining output protocols..... 467
Accessing existing application programs with ISC..... 468
Accessing programs that use MFS..... 468
Accessing programs that do not use MFS..... 469
Routing messages..... 469
Considerations for IMS-to-IMS ISC sessions..... 478
Statically defining an ISC node to IMS..... 480
Choosing parameters: system design considerations..... 482

COMM macro statement....482
NAME macro....483
SUBPOOL macro....483
TERMINAL macro....483
System definition summary....485

Chapter 30. ISC protocols for VTAM connections....487
Operating the network....487
Making IMS ready....487
Starting an IMS network for ISC....487
Shutting down an IMS network for ISC....487
Controlling the session (session control protocols)....488
Initiating an ISC VTAM session....488
Establishing a connection with an XRF complex....489
Binding the session....490
Resynchronizing sessions....491
Designing restart resynchronization procedures....492
Determining session synchronism using STSN....496
Performing the resynchronization....498
Completing session initiation....499
Running the session....499
Terminating an ISC VTAM session....499
Using STSN to resynchronize sessions....500
Primary-to-secondary flow matrix....500
Secondary-to-primary flow matrix....502
STSN command format....503
Handling IMS response mode or conversational output errors....505
Response mode errors....505
Conversational mode errors....506
Normal conversation termination extension with ISC....506
Keeping half sessions synchronized....506
Sync points requested on input to IMS....507
Sync points requested on output by IMS....508
Sync point and response requirements....508
Sync-point indicators on messages....510
Data flow control protocol reference....515
BID protocol....515
Bracket and half-duplex protocol....515
CANCEL protocol....523
Chaining protocol....524
CHASE protocol....524
ERP purging....525
LUSTATUS protocol....528
Paged messages ERP....531
Ready-to-receive protocol....531
RSHUT protocol....531
Selective receiver ERP....531
Sender ERP....536
Sense codes that IMS receives....538
Sense codes that IMS sends....539
SIGNAL protocol....540
Symmetrical session shutdown for LU 6.1 (SBI and BIS)....540
Function management headers....541
Using FM headers to invoke ISC edit....542
Initiating a process: ATTACH FM header....543
Error recovery procedure FM header....543
Resetting the active process: RAP FM header....544
Requesting asynchronous transaction processing: SCHEDULER FM header....544

System message process (SYSMSG) and related FM headers....545

Chapter 31. Using MFS with ISC....547
Activating MFS input formatting....547
Activating MFS output formatting for ISC....547
MFS Distributed Presentation Management (DPM) messages....548
MFS page delete function....548
MFS online error detection....548
The ATTACH and SCHEDULER FM headers under MFS....550
Data descriptor FM headers....551
Input data descriptor FM header....551
Output data descriptor FM header....551
Controlling demand-paged messages: QMODEL FM headers....552
Request (input) QMODEL FM headers....553
QGETN FM header....553
QGET FM header....553
QPURGE FM header....554
Reply (output) QMODEL FM headers....554
QXFR FM header....554
QSTATUS FM header....554
The RAP FM header under MFS....555

Chapter 32. FM header format reference....557
ATTACH FM header format....557
ATTIU....559
ATTDSP....560
ATTDBA....560
ATTDPN....561
ATTPRN....562
ATTRDPN and ATTRPRN....563
ATTDQN and ATTDP....564
ATTACC....565
Data descriptor FM header formats....565
Error recovery procedure (ERP) FM header....566
QMODEL FM headers....567
QGET FM header format....567
QGETN FM header format....568
QPURGE FM header format....569
QSTATUS FM header format....570
QXFR FM header format....571
Reset attached process (RAP) FM header format....573
SCHEDULER FM header format....573
SYSMSG process headers....575

Chapter 33. Examples using ISC edit ATTACH parameters....577
ATTACH and SCHEDULER parameters with ISC edit....577
ATTACH parameters with the IMS SYSMSG process....579
ATTACH and SCHEDULER parameters with IMS MFS....581

Chapter 34. How IMS and CICS use the ISC interface....587
Functions available to the ISC session....587
Overview of CICS synchronous and asynchronous processing for ISC....587
Functions available to an ISC TCP/IP session....589
Functions available to an ISC VTAM session....590
ISC communication with CICS over TCP/IP....593
Overview of ISC TCP/IP support....593
Requirements of ISC TCP/IP support....594
Restrictions for ISC TCP/IP support....595

Security for ISC TCP/IP connections....595
Setting up an ISC TCP/IP connection with CICS....596
Starting a session with CICS on an ISC TCP/IP link....601
Terminating an ISC TCP/IP session....603
Restarting an ISC TCP/IP session....604
CICS front-end transaction types supported by ISC over TCP/IP....604
General flow of CICS EXEC commands within a CICS application....605
CICS to IMS using SEND/RECEIVE EXEC commands....605
CICS to IMS using the SEND LAST EXEC command....607
IMS to CICS using the RECEIVE EXEC command....608
Coding asynchronous messages....609
CICS to IMS using the START/RETRIEVE EXEC commands....610
IMS to CICS using the RETRIEVE EXEC command....612
Commands that should not be used on an ISC session....612
Selecting appropriate CICS installation options for ISC....613
Coding CICS system definition options....613
Preparing CICS resource definition....613
Defining IMS-CICS LU 6.1 links....613
Defining compatible IMS and CICS nodes....614
System names....614
Number of sessions....615
Session names....615
Other session parameters....615
Defining multiple links to an IMS system....618
Defining CICS transactions for IMS-CICS ISC....620
Defining CICS backout in-doubt processing....620
Defining CICS transactions for asynchronous communication to IMS....620
Initiating and allocating a session from CICS....620
Other ways of initiating a session....621
Terminating a session from CICS....621
Designing CICS applications for ISC....622
Application-related concepts....622
Subsystem design: direct-control versus queued....622
Synchronous and asynchronous processing on ISC VTAM links....623
Principal and alternate facility....625
CICS versus IMS conversation mode....625
Sending IMS commands from CICS....626
Sync points....627
Coding function management headers for CICS....629
ATTACH function management header....629
SCHEDULER function management header....632
Queue model function management headers....634
Data descriptor function management header....635
System message process (SYSMSG) function management header....635
Error recovery procedure function management header....635
Recovery and restart concepts....635
Logical unit of work....636
Recovering outstanding message traffic after a failure....636
Handling transaction abends....639
Coding CICS applications for restart....640

Chapter 35. ISC data flow control examples....643
Non-MFS bracket and half-duplex protocol examples....643
MFS bracket and half-duplex protocol examples....644
MFS output examples....644
MFS input examples....648
SBI/BIS examples....649
Signal protocol example....651

Chapter 36. ISC error recovery procedure examples....653
Sender-detected error examples....653
Receiver-detected error examples....654

Chapter 37. Sample program for IMS-CICS ISC....657
Installation procedure....657
IMS sample program (DFSISC00)....658
Job control statements for the sample program....660
IMS system definition statements....661
MFS formats....662
Program specification block (PSB) generation for the sample program....663
Application control block (ACB) generation....664

Part 8. Multiple Systems Coupling (MSC)....665

Chapter 38. Overview of Multiple Systems Coupling....667
Multiple Systems Coupling concepts....667
MSC physical links....667
MSC logical links....669
MSC logical link paths....670
The MSC network and routing....671
Remote and local systems....671
Flow of data within multiple systems....672
Message routing....673
Routing path....673
Logical destinations....674
Input, destination, and intermediate systems....676
System identifiers (SYSIDs)....677
Routing messages with the destination name and SYSIDs....680
Remote LTERMs....681
Multiple Systems Coupling (MSC) directed routing....683
Remote destination verification....684

Chapter 39. Administering Multiple Systems Coupling....685
Design considerations for multiple systems....685
Minimizing resource consumption....685
Controlling the bandwidth of MSC links....686
Balancing resource demand....686
Planning for conversational processing....687
Routing exit routines with conversations....688
Remote destination verification for conversations....688
Saving truncated data in the SPA....688
Conversation termination....689
Abnormal conversation termination....689
Defining Multiple Systems Coupling resources....689
Enabling MSC in an IMS system....690
Disabling MSC with the MSC= execution parameter....693
Local system definitions....693
Defining partner systems....694
Defining the physical link....694
Defining the logical link....698
Defining a logical path....698
Setting link priorities for remote transactions....699
Serial transaction processing in an MSC network....701
Specifying exit routines....701
How network definition is affected by multiple systems....702

Verifying transaction definitions across systems....703
Using the multiple systems verification utility....703
Verifying the system definition status online....704
Security considerations for MSC....705
Operations for Multiple Systems Coupling....705
MSC link statistics....705
Benchmark link activity....706
Determine your optimum MSC link type....706
Reset statistics regularly at system checkpoint....706
Adjust link buffer sizes to the size of the messages....707
Adjust logical link capacity for MSC bandwidth mode....707
Determining optimum MSC link buffer sizes....707
Use high-value link statistics to help diagnose MSC link problems....710
Monitoring and tuning multiple systems....710
Coordinating performance information....711
Reports generated by the IMS Monitor for MSC....711
Extracting multiple-system transaction statistics....712
Controlling the log merge....712
Interpreting the Transaction Analysis report....712
MSC and IMSplexes with shared queues....713
Message routing across MSC and IMSplex environments....713
Migrating from an MSC network to an IMSplex network....715
Managing remote transactions for APPC and OTMA when MSC and IMSplexes coexist....719
Avoiding pseudoabend U0830....725
MSC TCP/IP generic resources....726
Managing MSC links in a TCP/IP generic resource group....726
Persistence of MSC link affinity in a TCP/IP generic resource group....728
Clearing MSC link affinity in a TCP/IP generic resource group....728
Clearing MSC link affinity in IMS Connect....729
XRF, MSC, and TCP/IP generic resources....729
VTAM Generic Resources (VGR) and MSC....730
TM and MSC Message Routing and Control user exit routine overview....730
The IMSplex affinity routing option of the DFSMSCE0 exit routine....732
Using the IMSRSC repository with MSC....733
IMSRSC repository definitions and MSC....733
How SIDR and SIDL values for remote trans and descriptors are stored....734
Maintaining MSC resources in the IMSRSC repository....734
Creating or updating MSC resources in the repository....737
Updating transactions from remote to local by using the repository....738
Updating transactions from local to remote by using the repository....738

Part 9. ODBA and DRA connections....741

Chapter 40. Accessing IMS databases with CICS....743
Coding considerations for PSBs....743
Using sequential buffering....744
CICS connected to DL/I....744
Configuring CICS CCTL connections to IMS DBCTL systems....744
CICS tasks....745

Chapter 41. Accessing IMS databases through the ODBA interface....747
Creating the ODBA DRA start-up table....747
Loading and running the ODBA and DRA modules in the z/OS application region....748
Binding application programs....748
Establishing and defining security....748
RAS security....749
Defining APSB security....750

Part 10. Open Transaction Manager Access (OTMA).....751

Chapter 42. Introduction to OTMA.....753
What is OTMA?.....753
Capabilities of OTMA.....754
Benefits of using OTMA.....755
Advantages of the OTMA protocol.....756
How IMS messages flow in an OTMA environment.....757
Basic OTMA message flow.....757
OTMA IMS-to-IMS TCP/IP communications flow.....759
Sample commit-then-send transaction processing flows.....760
Using transaction pipes with OTMA.....762
Differences in transaction pipes.....763
Message flow using transaction pipes.....763

Chapter 43. Enabling and using OTMA.....767
Enabling OTMA.....767
Summary of the OTMA configuration parameters.....767
Defining the XCF group name.....768
Defining the OTMA XCF member name.....768
Defining when OTMA starts up.....768
Defining the level of OTMA security checking.....769
OTMA tpipe support for parallel processing of multiple active RESUME TPIPE requests.....769
Specifying synchronized tpipes for IBM MQ.....772
Enabling OTMAYPRX member name override for OTMA clients.....772
Specifying asynchronous delivery of program-to-program switch output messages.....772
OTMA descriptors.....773
OTMA client descriptors.....773
OTMA destination descriptors.....774
DFSOTMA descriptor.....777
Changing the limits on OTMA descriptors.....778
OTMA support for IMS-to-IMS communications.....778
Super member support for IMS-to-IMS communications.....779
Specifying a remote transaction code.....780
Format of messages sent to a remote IMS system.....780
OTMA-supported exit routines.....781
Using the OTMAYPRX user exit and DFSYDRU0 exit routine to determine destination.....782
Administering IMS for OTMA.....784
IMS conversations and OTMA.....784
MSC and OTMA transactions.....784
Fast Path and OTMA transactions.....784
IMS restart processing and OTMA.....785
XRF processing and OTMA.....785
IMS Queue Control Facility and OTMA.....785
Using shared queues with OTMA.....786
IMS termination and OTMA.....791
OTMA client notification of IMS termination.....791
IMS termination and IMS-to-IMS TCP/IP messages.....791
OTMA restrictions and requirements.....791
Managing system resources and OTMA.....792
Administering OTMA tmembers.....792
Buffer pool usage for OTMA.....793
Collecting OTMA checkpoint statistics.....793
Dependent region occupancy and OTMA.....794
Displaying the current transaction workload.....794
Impact of OTMA message TIBs on storage.....795

IMS message queue data set size and OTMA....796
Message flood detection....796
Monitoring tpipe usage....797
Monitoring system resources with OTMA....799
OTMA ACEE flood control....800
Removing idle tpipes....800
Specifying acknowledgment timeout intervals for OTMA messages....801
Specifying an expiration time for transactions to OTMA....803
Specifying the number of SAPs IMS allocates for OTMA input messages....805
Terminating conversational transactions in OTMA....806
OTMA security....806
RACF security levels for OTMA....806
Specifying OTMA security....808
Securing messages on the asynchronous hold queue....813
Security for OTMA IMS-to-IMS TCP/IP connections....815
General OTMA security considerations....815
Using DL/I calls in an OTMA environment....816
OTMA program-to-program switch processing....817
OTMA single-stream program switch....818
OTMA program switch without ISRT to I/O PCB....818
OTMA program switch with express PCB....818
OTMA program switch to multiple programs....819
OTMA program switch for protected transactions....820
Other OTMA program switch considerations....820

Chapter 44. The OTMA client....821
What is an OTMA client?....821
OTMA naming conventions....822
Messages sent by OTMA clients....822
Sending type-1 commands from an OTMA client....824
OTMA commit processing....824
Summary of OTMA commit processing....825
Sample OTMA commit processing flows....826
Sample OTMA message flows....831
Protecting transactions with OTMA....836
Initiating protected transactions from an OTMA client....836
Processing protected transactions in IMS....836
Client/server resynchronization with OTMA....837
Assumptions for OTMA resynchronization....837
Recoverable OTMA transactions....838
Unrecoverable OTMA transactions....838
Summary results of IMS transactions and commands....838
OTMA resynchronization protocol....840
Sample OTMA resynchronization message flow....842
Sample OTMA resynchronization messages....844
Managing commit-then-send output....845
Purging commit-then-send asynchronous output....845
Rerouting commit-then-send asynchronous output....846
Timeout for acknowledgments of commit-then-send output....847
Sharing asynchronous commit-then-send output: the OTMA super member function....847
Displaying output on the asynchronous hold queue....848

Chapter 45. OTMA support for callout requests....849
Callout requests from IMS application programs....849
Synchronous callout requests....850
Synchronous program switch requests....851
Asynchronous callout request....853
Implementing the asynchronous callout function....853

IMS TM Resource Adapter and asynchronous callout requests..... 855
SOAP Gateway and asynchronous callout requests..... 855
IBM MQ and asynchronous callout requests..... 856
IMS application programs and the asynchronous callout function..... 856
Callout and OTMA parallel processing of RESUME TPIPE requests..... 858

Chapter 46. OTMA message prefix..... 859
Message-control information section..... 859
Explanation of OTMA message-control information fields..... 862
State data section..... 868
Server-Available and Client-Bid commands..... 868
SRVresynch command..... 872
REQresynch command..... 872
REPresynch command..... 872
TBresynch command..... 873
Transaction and callout messages..... 874
Server state protocol command..... 878
Resume output for Tpipe..... 880
Resume output for all Tpipes protocol command format..... 880
Resume output for the hold queue for tpipe..... 881
Cancel resume output for tpipe hold queue request..... 882
No messages on tpipe hold queue..... 882
Security data section..... 883
Explanation of OTMA security data fields..... 884
User data section..... 886
Explanation of OTMA user data fields..... 886
Application data section..... 886
Sample OTMA messages..... 887

Chapter 47. OTMA Callable Interface..... 889
OTMA C/I initialization..... 891
OTMA C/I security..... 892
OTMA C/I restrictions..... 892
Timing out OTMA C/I sessions after otma_send_receive API calls for CM1 transactions..... 892

Chapter 48. OTMA architected transaction attributes..... 895

Part 11. SLU P and Finance Communication..... 897

Chapter 49. Overview of SLU P and Finance Communication..... 899
The IMS-SLU P network..... 899
System configuration..... 900
SLU P and Finance workstations..... 900
System controller application program..... 900
Writing the controller application program with MFS and XRF..... 900
Considerations for controller application programs for XRF systems..... 901
Converting controller application programs from Finance to SLU P..... 901
VTAM facilities used..... 902
VTAM commands and indicators used with SLU P..... 902
Request-recovery command..... 904
Change-direction indicator..... 904
Establishing connection and specifying logon modes..... 904
Establishing connection with the XRF complex..... 905
Bracket and send/receive management..... 905

Chapter 50. IMS facilities used for SLU P and Finance..... 907
Component definition..... 907

LTERM naming ... 907
Output component selection ... 907
Input component determination ... 908
Terminal-response mode ... 908
Defining a workstation for terminal-response mode ... 909
Output messages sent while in a between-brackets state ... 910
Designing for output messages sent while in between-brackets state ... 911
IMS Message Format Service ... 911
Designing MFS for the workstation environment ... 911
MID/MOD chaining ... 912
MFS output formatting for the SLU P system ... 912
MFS message recovery ... 913
MFS control functions (Finance) ... 913
MFS control functions (SLU P) ... 913
MFS paging and BID options ... 914
Display screen protection for finance stations ... 914
Extended output component protection (SLU P) ... 914
Input and output editing options (SLU P) ... 916
Use of responses or brackets to acknowledge recoverable input ... 917
Message recovery ... 918
Message resynchronization ... 918
Finance and SLU P in an XRF complex ... 919
Fast Path messages with Finance and SLU P ... 919
Fast Path output messages (Finance) ... 920
Fast Path output messages (SLU P) ... 920
Fast Path message resynchronization ... 921

Chapter 51. Network operation for SLU P and Finance ... 923
Starting an IMS network ... 923
Making IMS ready ... 923
Session initiation (starting workstations) ... 923
Session-initiation transmission sequence ... 924
Controller application program involvement in message resynchronization ... 924
Design considerations ... 925
Sequence number management ... 925
Set-and-Test-Sequence-Numbers (STSN) ... 926
Suspending output from IMS ... 929
Session termination ... 929
Orderly termination ... 930
Immediate termination ... 930
Shutting down an IMS network (SLU P) ... 931
SLU P messages ... 931
Send/receive and bracket protocol ... 932

Chapter 52. SLU P message protocols ... 933
General format of input function management headers (Finance) ... 933
Input message descriptor byte (Finance) ... 933
General format of input function management headers (SLU P) ... 934
Input message descriptor bytes (SLU P) ... 935
Input component identification (SLU P) ... 935
Input bracketing protocol ... 936
Activating MFS input formatting for Finance workstations ... 936
Output messages ... 936
MFS Distributed Presentation Management output (SLU P) ... 939
General format of output function management headers (Finance) ... 939
Output message descriptor byte (Finance) ... 940
Output component ID byte (Finance) ... 940
MFS data bytes (Finance) ... 940

General format of output function management headers (SLU P)..... 940
Output message descriptor bytes (SLU P)..... 941
MFS data bytes (SLU P).....942
Output bracketing protocol..... 942
Activating MFS output formatting for SLU P..... 943
Response requests (Finance).....943
Response requests (SLU P)..... 944
Input response requirements.....944
Output response requirements..... 945
IMS transaction types..... 945
Recoverable-inquiry transactions..... 946
Irrecoverable-inquiry transactions..... 946
Verifying IMS receipt of irrecoverable messages..... 947
IMS message switches..... 947
IMS commands.....947
VTAM commands and indicators.....947
MFS control requests.....947
Error handling.....948
IMS-detected errors..... 948
Controller or station-detected errors.....949
VTAM logical unit status (LUSTATUS) command.....950
VTAM ready-to-receive (RTR) command..... 950
VTAM CANCEL command..... 951
VTAM request-recovery command.....951

Notices.....953
Programming interface information.....954
Trademarks.....954
Terms and conditions for product documentation..... 955
IBM Online Privacy Statement..... 955

Bibliography.....957

Index..... 959

About this information

These topics describe how to administer IMS communications and connections: CPI Communications and APPC/IMS, facilities for attaching to external subsystems, IMS Extended Terminal Option (ETO), IMS Connect, IMS Universal driver connections, Intersystem Communication (ISC), Multiple Systems Coupling (MSC), IMS Open Database Access (ODBA) and database resource adapter (DRA) interfaces, IMS Open Transaction Manager Access (OTMA), SLU P and Finance communication systems, TCP/IP communications, and VTAM® networking.

This information is available in IBM® Documentation.

Prerequisite knowledge

You should first read *IMS Version 15 System Administration*. Its introductory chapters, which cover planning activities, the IMS environments, and administration concepts, is particularly useful as a background for this book.

If you use APPC/IMS, you must be familiar with APPC/MVS in order to correctly define APPC/MVS configurations. For more information about APPC/MVS, see *CPI Communications Reference*.

If you implement ISC sessions between IMS and CICS®, you should understand the information in *CICS Transaction Server for z/OS CICS Intercommunication Guide*.

If you are connecting to IMS by using TCP/IP connections, you should understand z/OS® TCP/IP, as documented in z/OS Communications Server: IP Configuration Guide.

To learn about z/OS, see z/OS Basic Skills. For more resources, see IBM Z Education and Training.

To learn about IMS, see the IBM Press publication *An Introduction to IMS*, the resources listed for IBM Information Management System, and the variety of options available in IBM Training.

How new and changed information is identified

New and changed information in most IMS library PDF publications is denoted by a character (revision marker) in the left margin. The first edition (-00) of *Release Planning*, as well as the *Program Directory* and *Licensed Program Specifications*, do not include revision markers.

Revision markers follow these general conventions:

- Only technical changes are marked; style and grammatical changes are not marked.
- If part of an element, such as a paragraph, syntax diagram, list item, task step, or figure is changed, the entire element is marked with revision markers, even though only part of the element might have changed.
- If a topic is changed by more than 50%, the entire topic is marked with revision markers (so it might seem to be a new topic, even though it is not).

Revision markers do not necessarily indicate all the changes made to the information because deleted text and graphics cannot be marked with revision markers.

How to read syntax diagrams

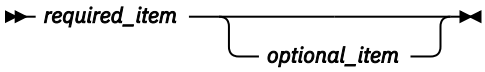
The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.

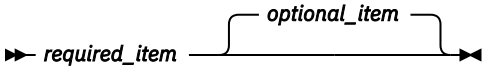
© Copyright IBM Corp. 1974, 2020

  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

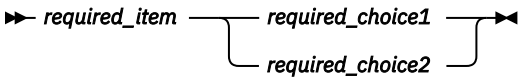


  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

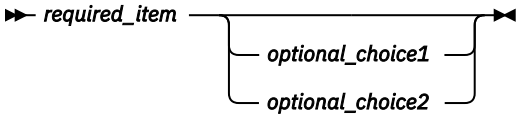


- If you can choose from two or more items, they appear vertically, in a stack.

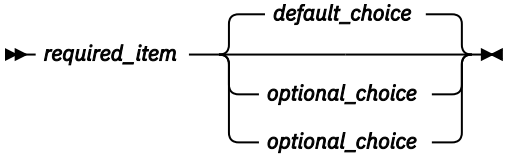
  If you *must* choose one of the items, one item of the stack appears on the main path.



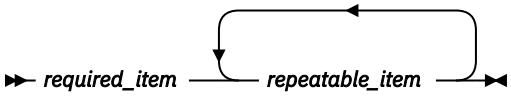
  If choosing one of the items is optional, the entire stack appears below the main path.



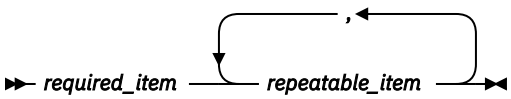
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



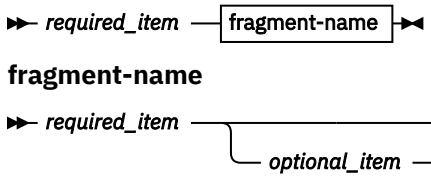
- An arrow returning to the left, above the main line, indicates an item that can be repeated.



  If the repeat arrow contains a comma, you must separate repeated items with a comma.



  A repeat arrow above a stack indicates that you can repeat the items in the stack.
- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



- In IMS, a b symbol indicates one blank position.
- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

Accessibility features for IMS 15

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in z/OS products, including IMS 15. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers and screen magnifiers.
- Customization of display attributes such as color, contrast, and font size.

Keyboard navigation

You can access IMS 15 ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the IMS 15 ISPF panels using TSO/E or ISPF, refer to the *z/OS TSO/E Primer*, the *z/OS TSO/E User's Guide*, and the *z/OS ISPF User's Guide Volume 1*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Related accessibility information

Online documentation for IMS 15 is available in IBM Documentation.

IBM and accessibility

See the *IBM Human Ability and Accessibility Center* at www.ibm.com/able for more information about the commitment that IBM has to accessibility.

How to send your comments

Your feedback is important in helping us provide the most accurate and highest quality information. If you have any comments about this or any other IMS information, you can take one of the following actions:

- Submit a comment by using the DISQUS commenting feature at the bottom of any IBM Documentation topic.
- Send an email to imspubs@us.ibm.com. Be sure to include the book title and the publication number.

- Click the **Contact Us** tab at the bottom of any IBM Documentation topic.

To help us respond quickly and accurately, please include as much information as you can about the content you are commenting on, where we can find it, and what your suggestions for improvement might be.

Part 1. Configuring external Java environment connections

The IMS Universal drivers support Java™ applications that access IMS. The IMS Universal drivers are built on industry standards and open specifications, and provide flexible support for connectivity, data access methods, and transaction processing options.

Related concepts

Transaction types and programming interfaces supported by the IMS Universal Database resource adapter (Application Programming)

IMS solutions for Java development overview (Application Programming)

© Copyright IBM Corp. 1974, 2020

Chapter 1. IMS Universal drivers: configuring connections to IMS

The IMS Universal drivers can run in z/OS and distributed environments, including WebSphere® Application Server for z/OS, WebSphere Application Server for distributed platforms, and WebSphere Application Server Liberty.

The IMS Universal drivers include the following adapters and drivers:

- The IMS Universal Database resource adapters, which take advantage of Java Platform, Enterprise Edition (Java EE) services.
- The IMS Universal JDBC driver, which support SQL calls that directly access your IMS data.
- The IMS Universal DL/I driver, which provides calls that are similar to DL/I in a Java programming interface.

When running in a distributed environment on a server such as WebSphere Application Server for distributed platforms, or in a remote z/OS environments on a server such as WebSphere Application Server for z/OS, or in WebSphere Application Server Liberty, the IMS Universal drivers connect to IMS using a type-4 connection architecture, which supports TCP/IP communications and socket management.

When running locally on the same logical partition (LPAR) as IMS, the IMS Universal drivers connect to IMS by using a type-2 connection architecture, which supports direct communication with IMS through the IMS Open Database Access (ODBA) and IMS database resource adapter (DRA) interfaces.

WebSphere Application Server supports all of the IMS Universal drivers in both distributed and z/OS environments.

Related concepts

Programming with the IMS Universal drivers (Application Programming)

Distributed and local connectivity with the IMS Universal drivers (Application Programming)

Related reference

Comparison of IMS Universal drivers programming approaches for accessing IMS (Application Programming)

Configuring WebSphere Application Server for EJB development with the IMS Universal drivers

To develop Enterprise JavaBeans (EJB) applications that run on WebSphere Application Server, on either z/OS or distributed platforms, custom properties for the Java Virtual Machine (JVM) and for XML/Java binding must be properly configured

To configure the WebSphere Application Server for EJB applications that use the IMS Universal drivers:

1. In the administrative console, select the appropriate Server instance and servant:
   a) Click **Servers** > **Application servers** > ***server_name***.
   b) In the Server Infrastructure section select **Java and process management** > **Process Definition**.
   c) Select **Servant**.
2. Set the Java Virtual Machine custom property **`com.ibm.ws.runtime.component.ResourceMgr.postBindNotify`** to `true`.
   a) Under Additional Properties, click **Java virtual machine**.
   b) Under Additional Properties, click **Custom Properties**.
   c) Set to `true` the custom property named **`com.ibm.ws.runtime.component.ResourceMgr.postBindNotify`**.

© Copyright IBM Corp. 1974, 2020

If the custom property is not present in the list of already defined custom properties, create it and set it to `true`.

3. If the IMS catalog is enabled in IMS and the **`javax.xml.bind.JAXBContext`** custom property does not already exist, take the following steps:
   a) Under Additional Properties, click **Java virtual machine**.
   b) Under Additional Properties, click **Custom Properties**.
   c) Create a custom property named **`javax.xml.bind.JAXBContext`** and set it to `com.sun.xml.internal.bind.v2.ContextFactory`.
4. Save the changes.
5. Restart the server.

You can query the custom properties of the corresponding Connection Factory, use the getProperty method exposed by the Connection Factory MBean.

IMS Universal drivers: WebSphere Application Server type-4 connections

Java applications that run on WebSphere Application Server can access IMS databases by using the type-4 connectivity provided by the IMS Universal drivers. The type-4 connectivity of the IMS Universal drivers enables Java application programs to access IMS databases from a wide variety of distributed and mainframe environments, either in stand-alone mode or under an application server, with or without XA support for global transactions.

The following figure provides an overview of a configuration that uses the type-4 connectivity of an IMS Universal Database resource adapter to connect to IMS from WebSphere Application Server for distributed platforms.

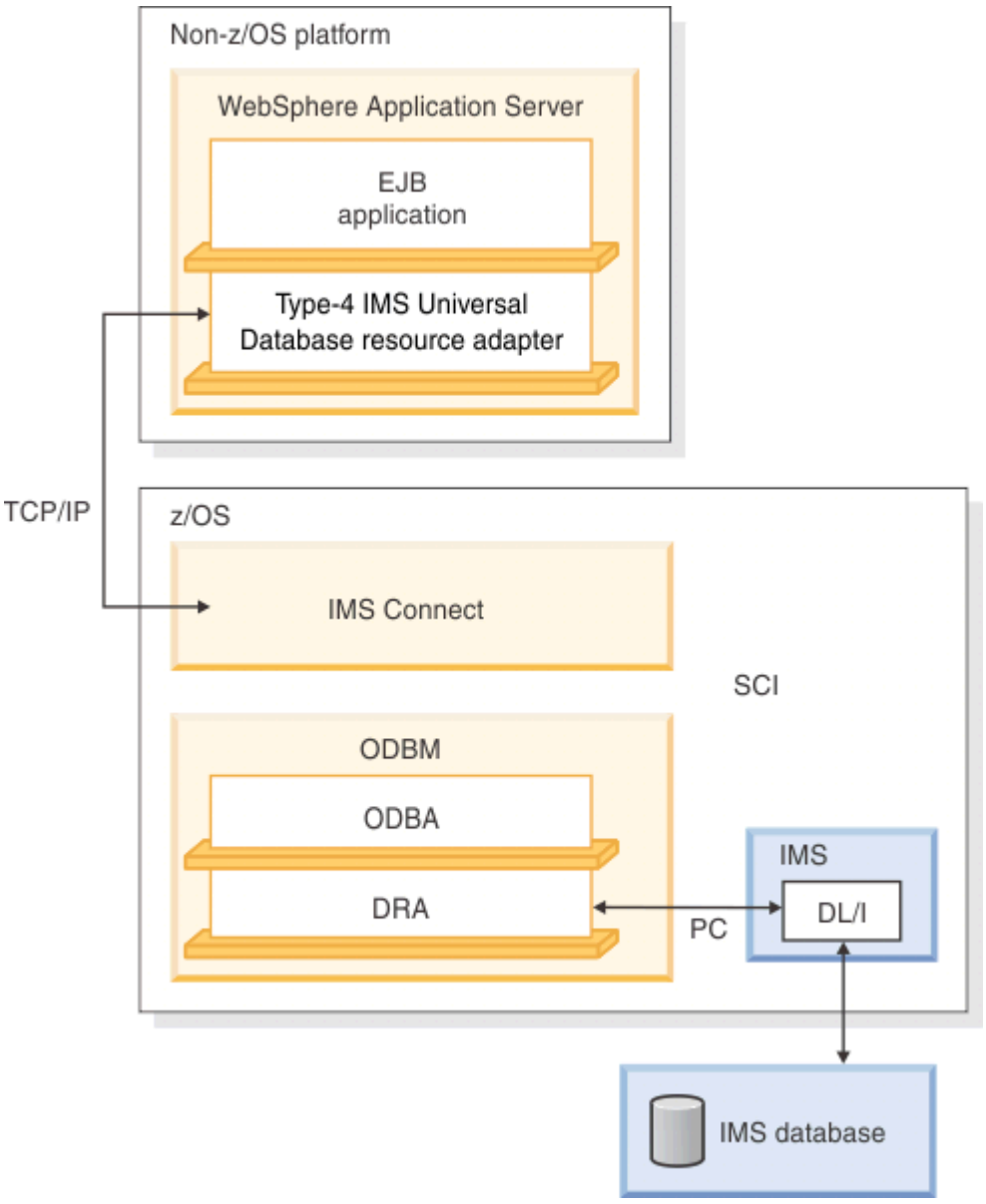


Figure 1. A WebSphere Application Server EJB application using a type-4 IMS Universal Database resource adapter

The following procedures apply to both WebSphere Application Server for distributed platforms and WebSphere Application Server for z/OS.

Related concepts

Transaction types and programming interfaces supported by the IMS Universal Database resource adapter (Application Programming)

Installing a type-4 IMS Universal Database resource adapter on WebSphere Application Server

You must install an IMS Universal Database resource adapter before it can be used to access IMS databases from WebSphere Application Server.

Prerequisites:

- You must configure your WebSphere Application Server by following the steps in "Configuring WebSphere Application Server for EJB development with the IMS Universal drivers" on page 3.

- Copy the RAR files for the IMS Universal drivers to storage that is accessible to WebSphere Application Server.

To install the IMS Universal Database resource adapter:

1. In the WebSphere Application Server administrative console, click **Resources** > **Resource Adapters** > **Resource Adapters**.
2. Click **Install RAR**.

   A dialog opens for installing the resource adapter.
3. Enter the path to the RAR files. The path can be to a local location or to a location on the server.
   - If your RAR file is located on your local workstation, select **Local path** and browse to find the file. For example: `C:\`*`install_directory`*`\imsudbLocal.rar`
   - If your RAR file is located on your server, select **Server path** and specify the fully qualified path to the file.
4. Click **Next**.

   The configuration panel opens.
5. Click **OK**.

   The IMS Universal Database resource adapter is listed.
6. In the messages box, click **Save**.

   The save page is displayed.
7. Click **Save** to update the master repository with your changes.

Related reference

Comparison of IMS Universal drivers programming approaches for accessing IMS (Application Programming)

Defining a connection factory for a type-4 IMS Universal Database resource adapter on WebSphere Application Server

After you install an IMS Universal Database resource adapter, you define the connection factory by using the WebSphere Application Server administrative console.

To define the connection factory for the IMS Universal Database resource adapter:

1. In the left frame of the WebSphere Application Server administrative console, click **Resources** > **Resource Adapters** > **Resource Adapters**.
2. Click the name of the IMS Universal Database resource adapter that you chose when you installed the adapter.
3. Under Additional Properties, click **J2C connection factories**.
4. Click **New**.
5. Type the following information:

   Name: the name for the connection factory, for example, PhonebookCF

   JNDI Name: the JNDI name for the connection factory that is unique within this server, for example, PhonebookCF
6. Click **Apply**.

   The data source is listed in the J2C Connection Factories.
7. Under Additional Properties, click **Custom Properties**.
8. Specify values defined by your installation for each of the following properties by clicking the property name and, in the configuration pane that opens, specifying the value for the property in the **Value** field. After specifying the value for a property, click OK to return to the list of properties.

   See Connecting using the IMS Universal Database resource adapter in a managed environment (Application Programming) for a list of connection properties.
9. In the messages box, click **Save**.

10. Restart WebSphere Application Server.

Related concepts

IMS Connect definition and tailoring (System Definition)

Installing an EAR file that uses a type-4 IMS Universal Database resource adapter on WebSphere Application Server

After installing a type-4 IMS Universal Database resource adapter and configuring the connection factory, you must install the client application program, or EAR file, on WebSphere Application Server.

To install the EAR file on the WebSphere Application Server:

1. In the left pane of the WebSphere Application Server administrative console, click **Applications** > **Install New Application**.
2. Enter the path to the EAR file or locate the EAR file by clicking the **Browse** button. Click **Next**.
3. In the "Selection installation options" panel, accept the defaults and click **Next**.
4. In the "Map modules to servers" panel, accept the defaults and click **Next**.
5. In the "Summary" panel, verify that the options are correct and click **Next**.

   A series of messages are issued that indicate that the application is being installed.
6. After the message that indicates that your application was installed successfully, click **Save**.

After your application is installed, it is ready to be started.

Related concepts

Programming with the IMS Universal drivers (Application Programming)

IMS Universal drivers: WebSphere Application Server for z/OS type-2 connections

When WebSphere Application Server for z/OS and IMS are on the same logical partition (LPAR), Java applications running in WebSphere Application Server for z/OS can access IMS databases by using the type-2 connectivity provided by the IMS Universal Database resource adapters.

The type-2 connectivity of the IMS Universal Database resource adapters provides local, non-TCP/IP access to IMS databases.

To deploy an application that uses the type-2 connectivity, you must install one of the IMS Universal Database resource adapters in WebSphere Application Server for z/OS and configure the IMS Open Database Access (ODBA) interface.

IMS Universal Database resource adapter type-2 connectivity supports both bean-managed bean methods and container-managed bean methods. Optionally, transaction applications can be managed by the z/OS Resource Recovery Services local option.

The following figure shows an EJB application that is accessing IMS data. The database requests are passed to a type-2 IMS Universal Database resource adapter, which converts the requests to DL/I calls. The IMS Universal Database resource adapter passes these calls to ODBA, which uses the IMS database resource adapter (DRA) interface to access the DL/I region in IMS.

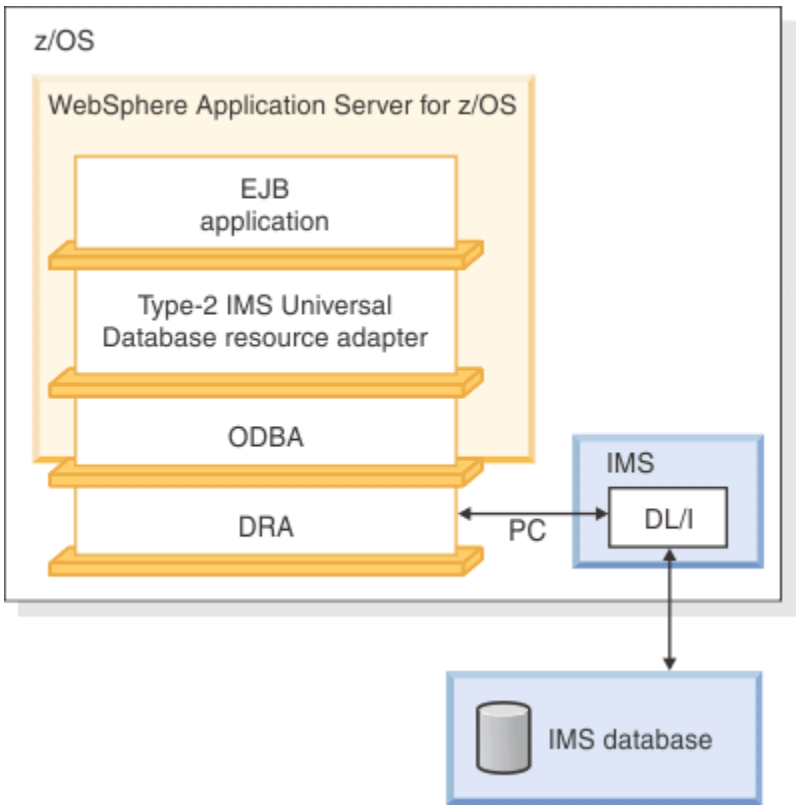


Figure 2. A WebSphere Application Server for z/OS EJB application using a type-2 IMS Universal Database resource adapter

Installing a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS

After you configure WebSphere Application Server for z/OS to have access to IMS databases, you must install the type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS.

Prerequisites:

1. Perform the steps described in "Configuring WebSphere Application Server for EJB development with the IMS Universal drivers" on page 3.
2. Copy the RAR files for the IMS Universal drivers to storage that is accessible to WebSphere Application Server for z/OS.
3. If not already done, create an ODBA startup table. The ODBA startup table module name can be from 5 to 8 bytes long and must conform to the following naming convention:
   - Bytes 1-3 must be "DFS"
   - Bytes 4-7 are the 1- to 4-byte ID
   - The final byte must be the character number "0"

   For example, both DFS10 and DFSIMSA0 are valid module names for an ODBA startup table.

   Recommendation: Use the IMS ID as the 1- to 4-byte ID.
4. If not already done, link the ODBA startup table into a load library.
5. Update the JCL for WebSphere Application Server for z/OS by adding to the STEPLIB the following data sets:
   - The load library that contains the ODBA startup table and the ODBA runtime code.
   - The SDFSJLIB data set. This data set contains the DFSCLIB member.

6. Note the ODBA name, which is defined by the MBR parameter. You will need to know bytes 4-7, which are usually the IMS system ID, when you install the data source.

To install the type-2 IMS Universal Database resource adapter:

1. In the WebSphere Application Server for z/OS administrative console, click **Resources** > **Resource Adapters** > **Resource Adapters**.

   A list of resource adapters is displayed.
2. Click **Install RAR**.

   A panel is displayed for installing the resource adapter.
3. In the "Install RAR" panel:
   - Under **Scope**, select a node
   - Under **Path**, enter the path to the RAR file or locate the RAR file by clicking the **Browse** button
4. Click **Next**.

   A configuration panel opens.
5. In the "Configuration" panel under **General properties** > **Native library path**, enter the path to the directory that contains the libT2DLI.so file and click **OK**.

   The libT2DLI.so file must have the proper read and execute permissions in Unix Systems Services. Also the SDFSJLIB must be included in the STEPLIB for the WebSphere servant region.

   After you click **OK**, the IMS Universal Database resource adapter that you installed is listed.
6. In the messages box, click **Save**.

   A page is displayed that asks you if you want to synchronize the changes with the nodes.
7. Click **OK** to update the master repository with your changes.

Related concepts

Database resource adapter (DRA) (System Programming APIs)

Related tasks

“Accessing IMS databases through the ODBA interface” on page 747

Open Database Access (ODBA) provides a callable interface that enables any z/OS recoverable, resource-managed z/OS address space to issue DL/I database calls to an IMS DB subsystem.

Optional: set the WebSphere Application Server for z/OS classpath for applications that use a type-2 IMS Universal Database resource adapter

Your application can include the IMS database metadata class (the `DLIDatabaseView` subclass that is generated by the IMS Enterprise Suite Explorer for Development) or the metadata class can be stored elsewhere.

If your application does not include the metadata class, you must set the WebSphere Application Server for z/OS classpath to point to the IMS database metadata class that is used by the application.

One way to set the classpath is to add these files to the classpath of your IMS Universal Database resource adapter.

To add the required files to the classpath of an IMS Universal Database resource adapter:

1. From the WebSphere Application Server for z/OS administrative console, click **Resources** > **Resource Adapters**.

   A list of resource adapters is displayed.
2. Click the name of your IMS Universal Database resource adapter.

   A configuration dialog is displayed.
3. In the class path text box, add the location of the metadata class.
4. Click **OK**.
5. In the messages box, click **Save**.

The save page is displayed.

6. Click **Save** to update the master repository with your changes.

Related reference

Generating the runtime Java metadata class (Application Programming)

Defining a connection factory for a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS

The `DataSource` facility is a factory for connections to a physical data source, or database. A data source is registered with a naming service based on the Java Naming and Directory (JNDI) API. `DataSource` objects have properties that pertain to the actual data source that an application needs to access.

Requirement: You must use the `DataSource` facility, which replaces the `DriverManager` facility, because the `DriverManager` facility is not supported by the Java EE Connection Architecture Specification.

To install the data source for your application:

1. In the left frame of the WebSphere Application Server for z/OS administrative console, click **Resources** > **Resource Adapters** > **Resource Adapters**.

   A list of resource adapters is displayed.
2. Click the name of the IMS Universal Database resource adapter that you chose when you installed the adapter.

   A configuration dialog is displayed.
3. Under Additional Properties, click **J2C connection factories**.
4. Click **New**.

   A configuration dialog is displayed.
5. Under General Properties, type the following information:

   Name: the name for the connection factory, for example, PhonebookCF.
   JNDI Name: the JNDI name for the connection factory that is unique within this server, for example, PhonebookCF.
6. Click **OK**.

   The new connection factory is listed in the table of resources that you can administer.
7. Click the name of the data source that you just installed.
8. Under Additional Properties, click **Custom Properties**.

   The properties are listed in a table.
9. Specify values defined by your installation for each of the following properties by clicking the property name and, in the configuration pane that opens, specifying the value for the property in the **Value** field. After specifying the value for a property, click OK to return to the list of properties.

   DatastoreName
   Enter bytes 4-7 of the DRA startup table module name (usually the IMS system ID). For more information about the DRA startup table, see "Installing a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS" on page 8.

   DriverType
   Set the driverType to either of the following values:

   2
   Specifies a local transaction model in which a unit of work is scoped to a particular connection. Multiple connections can have independent units of work associated with each.

   Application programs can issue local commit and rollback calls through either the JDBC Connection interface or the CCI LocalTransaction interface.

   DriverType=2 does not support the UserTransaction interface.

Container-managed bean methods require the following properties in the EJB Deployment Descriptor:

- In the **Bean** tab, specify the following properties under the **LocalTransaction** heading:
  - **Boundary:** `BeanMethod`
  - **Resolver:** `ContainerAtBoundary`
  - **Unresolved action:** `Rollback`
- In the **Assembly** tab, set the transaction scope to `NotSupported`.

2_CTX

Specifies a global scope transaction model in which a unit of work can span multiple bean methods.

Application programs can use the UserTransaction interface for explicit commit and rollback calls.

Application programs cannot issue local commit and rollback calls through either the JDBC Connection interface or the CCI LocalTransaction interface.

When 2_CTX is specified, use the default properties of the EJB Deployment Descriptor.

DatabaseName

Enter the location of the database metadata representing the target IMS database. You can specify this property in one of the following ways:

- The name of the PSB that is used to access the target database. This option is only available if your IMS system uses the IMS catalog.
- The fully qualified name of the Java metadata class generated by the IMS Enterprise Suite Explorer for Development. The URL must be prefixed with class:// (for example, `class://com.foo.BMP255DatabaseView`).

10. In the messages box, click **Save**.

    The save page is displayed.
11. Click **Save** to update the master repository with your changes.
12. Restart the server.

Installing an EAR file that uses a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS

This topic describes how to deploy an application on WebSphere Application Server for z/OS.

Prerequisite: Perform the steps described in "Defining a connection factory for a type-2 IMS Universal Database resource adapter on WebSphere Application Server for z/OS" on page 10.

To install your application:

1. In the left pane of the WebSphere Application Server for z/OS administrative console, click **Applications** > **New Application** > **New Enterprise Application**.
2. Enter the path to the EAR file or locate the EAR file by clicking the **Browse** button. Click **Next**.
3. In the "Preparing for the application installation" panel, accept the default and click **Next**.
4. In the "Selection installation options" panel, accept the defaults and click **Next**.
5. In the "Map modules to servers" panel, accept the defaults and click **Next**.
6. In the "Summary" panel, verify that the options are correct and click **Next**.

   A series of messages are issued that indicate that the application is being installed.
7. After the message that indicates that your application was installed successfully, click **Save**.

IMS Universal drivers: WebSphere Application Server Liberty type-4 connections

Java applications that run on WebSphere Application Server Liberty can access IMS databases by using the type-4 connectivity provided by IMS Universal Database resource adapters.

The type-4 connectivity of the IMS Universal drivers enables Java application programs to access IMS databases from a wide variety of distributed and mainframe environments, either in stand-alone mode or under an application server, with or without XA support for global transactions.

The following figure provides an overview of an EJB application that uses the type-4 connectivity of an IMS Universal Database resource adapter to connect to an IMS database from WebSphere Application Server for distributed platforms. Database requests are passed to a type-4 IMS Universal Database resource adapter, sent via TCP/IP to IMS Connect, and internally managed by ODBM to access IMS databases.

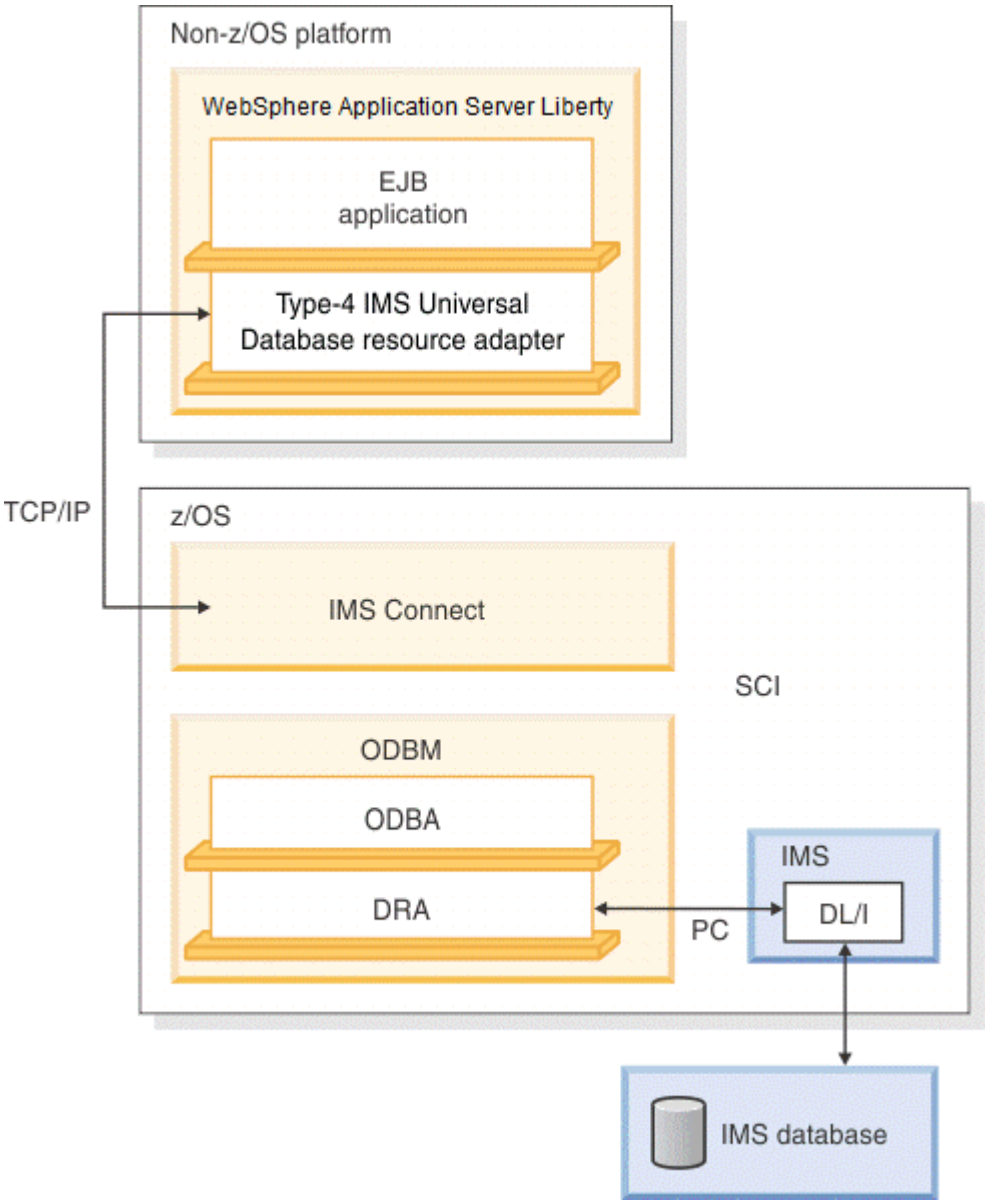


Figure 3. A WebSphere Application Server Liberty EJB application using a type-4 IMS Universal Database resource adapter

To enable Enterprise JavaBeans (EJB) applications that run on WebSphere Application Server Liberty, on a distributed platform, the `server.xml` configuration file must first be configured. It is used to install IMS Universal Database resource adapters, define connection factories that connect to an IMS™ database, and deploy RESTful service applications in WebSphere Application Server Liberty.

Enable Features

Within the <featureManager> element, add the <features> tags listed in the following sample to enable JCA and JDBC support for Liberty.

```
<featureManager>
    <feature>jca-1.7</feature>
    <feature>jndi-1.0</feature>
    <feature>jdbc-4.1</feature>
    <feature>localConnector-1.0</feature>
</featureManager>
```

Specify the IMS Universal Drivers Library locations

Within the <library> element, add the <fileset> element that points to the IMS Universal drivers library.

```
<library id="global">
    <!-- Include imsudb.jar -->
    <fileset dir="usr/lpp/…imsjava/" includes="imsudb.jar"/>
</library>
```

Install IMS Universal Database Resource Adapters

Install a resource adapter by adding the <resourceAdapter> element and defining its lookup `id` and resource `location` properties.

```
<resourceAdapter id="imsudbJLocal"
    location="usr/lpp/…imsjava/rar/imsudbJLocal.rar"/>
<resourceAdapter id="imsudbLocal"
    location="usr/lpp/…imsjava/rar/imsudbLocal.rar"/>
<resourceAdapter id="imsudbJXA"
    location="usr/lpp/…imsjava/rar/imsudbJXA.rar"/>
<resourceAdapter id="imsudbXA"
    location="usr/lpp/…imsjava/rar/imsudbXA.rar"/>
```

Define Connection Factories

Associate a connection factory to a resource adapter by adding a <connectionFactory> element and selecting the appropriate resource adapter by defining the `properties` subelement with the appropriate resource adapter `id`.

```
<!-- Associate a connection factory to a resource adapter
through the resource adapter's id. -->
<connectionFactory jndiName="HOSP_JDBC_T4">
    <properties.imsudbJLocal databaseName="MYPSB"
    datastoreName="IMS1" datastoreServer="myHostName"
    portNumber="1234" driverType="4"
    user="myUserID" password="myPassword"/>
</connectionFactory>
<connectionFactory jndiName="HOSP_CCI_T4">
    <properties.imsudbLocal databaseName="MYPSB"
    datastoreName="IMS1" datastoreServer="myHostName"
    portNumber="1234" driverType="4"
    user="myUserID" password="myPassword"/>
</connectionFactory>
<connectionFactory jndiName="HOSP_JDBC_T4XA">
    <properties.imsudbJXA databaseName="MYPSB"
    datastoreName="IMS1" datastoreServer="myHostName"
    portNumber="1234" driverType="4"
    user="myUserID" password="myPassword"/>
</connectionFactory>
<connectionFactory jndiName="HOSP_CCI_T4XA">
    <properties.imsudbXA databaseName="MYPSB"
    datastoreName="IMS1" datastoreServer="myHostName"
    portNumber="1234" driverType="4"
    user="myUserID" password="myPassword"/>
</connectionFactory>
```

Enable Trace and Logging

Trace and logging are disabled by default. Add the following <logging> element that contains the `traceSpecification` attribute with the String argument "com.ibm.ims.db.opendb" as follows to enable IMS Universal drivers trace and logging.

```
<!-- The log files can be found at: [usr/lpp/…/servers/server_name/logs] -->
<logging traceSpecification="*=info:com.ibm.ims.db.opendb.*=finest"/>
```

WebSphere Application Server Liberty type-4 connections sample server.xml configuration file

The sample server.xml file is used to install IMS Universal Database resource adapters, define connection factories which connect to an IMS™ database, and deploy an RESTful service application in WebSphere Application Server Liberty.

```
<server description="new server">

    <featureManager>
        <feature>jca-1.7</feature>
        <feature>jndi-1.0</feature>
        <feature>jdbc-4.1</feature>
        <feature>localConnector-1.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host
    attribute to the following element, e.g. host="*" -->
    <httpEndpoint host="*" httpPort="1692" httpsPort="9443"
        id="defaultHttpEndpoint"/>

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <library id="global">
        <!-- Include imsudb.jar -->
        <fileset dir="usr/lpp/…imsjava/" includes="imsudb.jar"/>

        <!-- (OPTIONAL) Include jars that contain local database
        metadata (dbviews) -->
        <fileset dir="usr/lpp/…/" includes="dbviews.jar"/>
    </library>

    <!-- Defining resource adapters -->
    <resourceAdapter id="imsudbJLocal"
        location="usr/lpp/…imsjava/rar/imsudbJLocal.rar"/>
    <resourceAdapter id="imsudbLocal"
        location="usr/lpp/…imsjava/rar/imsudbLocal.rar"/>
    <resourceAdapter id="imsudbJXA"
        location="usr/lpp/…imsjava/rar/imsudbJXA.rar"/>
    <resourceAdapter id="imsudbXA"
        location="usr/lpp/…imsjava/rar/imsudbXA.rar"/>

    <!-- Defining connection factories -->
    <!-- Associate a connection factory to a resource adapter through
    the resource adapter's id. -->
    <connectionFactory jndiName="HOSP_JDBC_T4">
        <properties.imsudbJLocal databaseName="MYPSB"
        datastoreName="IMS1" datastoreServer="myHostName"
        portNumber="1234" driverType="4"
        user="myUserID" password="myPassword"/>
    </connectionFactory>
    <connectionFactory jndiName="HOSP_CCI_T4">
        <properties.imsudbLocal databaseName="MYPSB"
        datastoreName="IMS1" datastoreServer="myHostName"
        portNumber="1234" driverType="4"
        user="myUserID" password="myPassword"/>
    </connectionFactory>
    <connectionFactory jndiName="HOSP_JDBC_T4XA">
        <properties.imsudbJXA databaseName="MYPSB"
        datastoreName="IMS1" datastoreServer="myHostName"
        portNumber="1234" driverType="4"
        user="myUserID" password="myPassword"/>
    </connectionFactory>
    <connectionFactory jndiName="HOSP_CCI_T4XA">
        <properties.imsudbXA databaseName="MYPSB"
```

```
        datastoreName="IMS1" datastoreServer="myHostName"
        portNumber="1234" driverType="4"
        user="myUserID" password="myPassword"/>
    </connectionFactory>

    <!-- Enable or Disable (Default) JDBC trace -->
    <!-- The log files can be found at:
          [usr/lpp/…/servers/server_name/logs] -->
    <logging traceSpecification=
        "*=info:com.ibm.ims.db.opendb.*=finest"/>

    <webApplication id="myApp" location="myApp.war" name="myApp"/>

</server>
```

IMS Universal drivers: WebSphere Application Server Liberty type-2 connections

When WebSphere Application Server Liberty and IMS are on the same logical partition (LPAR), the Java applications that run on WebSphere Application Server Liberty can access IMS databases by using the type-2 connectivity that is provided by the IMS Universal drivers.

The type-2 connectivity of the IMS Universal drivers enables Java application programs to access local, non-TCP/IP IMS databases.

With type-2 connections, WebSphere Application Server Liberty can be set up to access IMS databases through ODBM or through ODBA. In either case, Resource Recovery Services (RRS) must be active (RRS=Y) for both IMS and ODBA. To learn more about ODBA and RRS settings, see Chapter 41, “Accessing IMS databases through the ODBA interface,” on page 747 and CSL ODBM administration (System Administration).

ODBM is the recommended setup option for most use cases because it provides increased failure isolation and reduces the possibility of a U113 abend. In the following figure, an Enterprise JavaBeans (EJB) application running on WebSphere Application Server Liberty accesses an IMS database by passing requests to ODBM through a type-2 IMS Universal Database resource adapter.

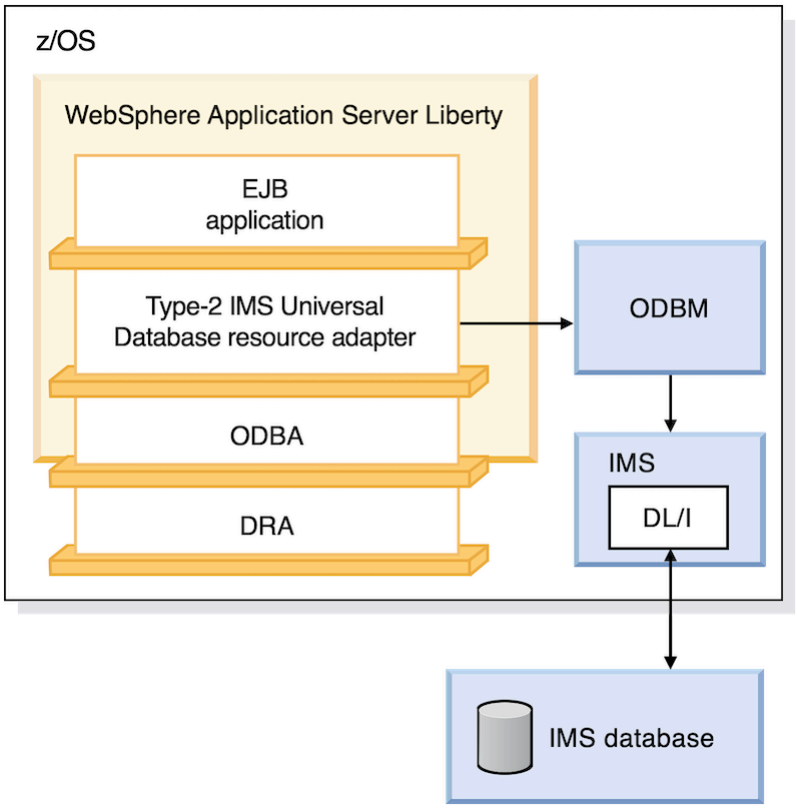


Figure 4. A WebSphere Application Server Liberty EJB application using a type-2 IMS Universal Database resource adapter with ODBM

Using ODBA and DRA is also a valid option. In the following figure, an Enterprise JavaBeans (EJB) application running on WebSphere Application Server Liberty accesses an IMS database by passing requests to a type-2 IMS Universal Database resource adapter. The requests are converted to DL/I calls and passed to ODBA. ODBA uses the IMS DRA interface to access the DL/I region in IMS.

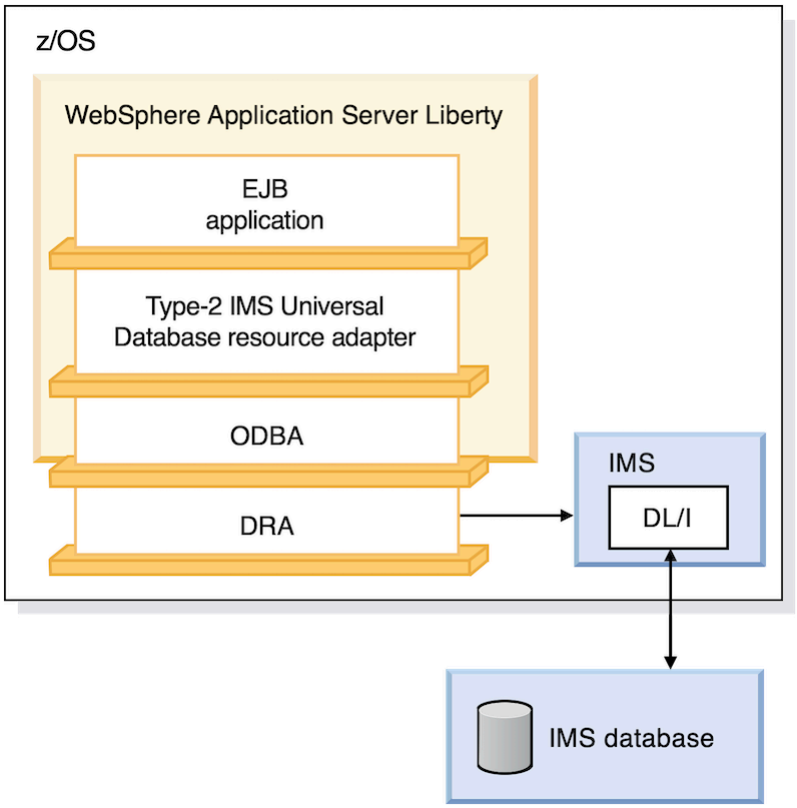


Figure 5. A WebSphere Application Server Liberty EJB application using a type-2 IMS Universal Database resource adapter with ODBA and DRA

To enable EJB applications that run on WebSphere Application Server Liberty on a z/OS platform, the `server.xml` configuration file must first be configured. It is used to install IMS Universal Database resource adapters, define connection factories that connect to an IMS database, and deploy RESTful service applications in WebSphere Application Server Liberty.

Enable Features

Within the <featureManager> element, add the <features> listed in the following sample to enable JCA and JDBC support for Liberty. The `zosTransaction-1.0` feature is required for RRS Type-2 connectivity.

```
<featureManager>
    <feature>jca-1.7</feature>
    <feature>jndi-1.0</feature>
    <feature>jdbc-4.1</feature>
    <feature>localConnector-1.0</feature>

    <!-- Required for RRS Type-2 connectivity -->
    <feature>zosTransaction-1.0</feature>
</featureManager>
```

Specify the IMS Universal Drivers Library locations

Within the <library> element, add the <fileset> subelements that point to the IMS Universal drivers libraries.

```
<library id="global">
    <!-- Include imsudb.jar -->
    <fileset dir="usr/lpp/…imsjava/" includes="imsudb.jar"/>

    <!-- libT2DLI.so or libT2DLI_64.so native code required for Type-2 Connectivity -->
    <fileset dir="usr/lpp/…/" includes="libT2DLI.so"/>
</library>
```

Enable Native Transaction Manager

Add the following <nativeTransactionManager> element for RRS Type-2 connectivity support.

```
<resourceAdapter id="imsudbJLocal"
location="usr/lpp/…imsjava/rar/imsudbJLocal.rar"/>
<resourceAdapter id="imsudbLocal"
location="usr/lpp/…imsjava/rar/imsudbLocal.rar"/>
```

Define Connection Factories

Associate a connection factory to a resource adapter by adding a <connectionFactory> element and selecting the appropriate resource adapter by defining the `properties` subelement with the appropriate resource adapter `id`.

```
<!-- Associate a connection factory to a resource adapter
through the resource adapter's id. -->
<connectionFactory jndiName="HOSP_JDBC_T2">
        <properties.imsudbJLocal databaseName="MYPSB"
        datastoreName="IMS1" driverType="2"/>
</connectionFactory>
<connectionFactory jndiName="HOSP_CCI_T2">
        <properties.imsudbLocal databaseName="MYPSB"
        datastoreName="IMS1" driverType="2"/>
</connectionFactory>
```

Enable Trace / Logging

Trace and logging are disabled by default. Add the following <logging> element that contains the `traceSpecification` attribute with the String argument "com.ibm.ims.db.opendb" as follows to enable IMS Universal drivers trace and logging.

```
<!-- The log files can be found at: [usr/lpp/…/servers/server_name/logs] -->
<logging traceSpecification="*=info:com.ibm.ims.db.opendb.*=finest"/>
```

WebSphere Application Server Liberty type-2 connections sample server.xml configuration file

The sample server.xml file is used to install IMS Universal Database resource adapters, define connection factories which connect to an IMS™ database, and deploy an RESTful service application in WebSphere Application Server Liberty.

```
<server description="new server">

    <featureManager>
        <feature>jca-1.7</feature>
        <feature>jndi-1.0</feature>
        <feature>jdbc-4.1</feature>
        <feature>localConnector-1.0</feature>

        <!-- Required for RRS Type-2 connectivity -->
        <feature>zosTransaction-1.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host
    attribute to the following element, e.g. host="*" -->
    <httpEndpoint host="*" httpPort="1692" httpsPort="9443"
        id="defaultHttpEndpoint"/>

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <library id="global">
        <!-- Include imsudb.jar -->
        <fileset dir="usr/lpp/…imsjava/" includes="imsudb.jar"/>

        <!-- libT2DLI.so or libT2DLI_64.so native code required for
        Type-2 Connectivity -->
        <fileset dir="usr/lpp/…/" includes="libT2DLI.so"/>

        <!-- (OPTIONAL) Include jars that contain local database
        metadata (dbviews) -->
        <fileset dir="usr/lpp/…/" includes="dbviews.jar"/>
    </library>
```

```
    <nativeTransactionManager shutdownTimeout="5s"/>

    <!-- Defining resource adapters -->
    <resourceAdapter id="imsudbJLocal"
        location="usr/lpp/…imsjava/rar/imsudbJLocal.rar"/>
    <resourceAdapter id="imsudbLocal"
        location="usr/lpp/…imsjava/rar/imsudbLocal.rar"/>

    <!-- Defining connection factories -->
    <!-- Associate a connection factory to a resource adapter through
    the resource adapter's id. -->
        <connectionFactory jndiName="HOSP_JDBC_T2">
        <properties.imsudbJLocal databaseName="MYPSB"
        datastoreName="IMS1" driverType="2"/>
    </connectionFactory>
    <connectionFactory jndiName="HOSP_CCI_T2">
        <properties.imsudbLocal databaseName="MYPSB"
        datastoreName="IMS1" driverType="2"/>
    </connectionFactory>

    <!-- Enable or Disable (Default) JDBC trace -->
    <!-- The log files can be found at:
        [usr/lpp/…/servers/server_name/logs] -->
    <logging traceSpecification=
    "*=info:com.ibm.ims.db.opendb.*=finest"/>

    <webApplication id="myApp" location="myApp.war" name="myApp"/>

</server>
```

The IMS Universal drivers: CICS connections

Java applications that run on IBM CICS Transaction Server for z/OS can access IMS databases by using the type-2 connectivity provided by the IMS Universal drivers. Other than the Java layer, access to IMS from a Java application is the same as for a non-Java application.

Note: Type-2 connectivity support will be delivered through the IMS service process.

CICS supports the following IMS Universal drivers:

- IMS Universal JDBC driver
- IMS Universal DL/I driver

The following figure shows a JCICS application that is accessing an IMS database by using the IMS database resource adapter (DRA) interface and the type-2 connectivity of an IMS Universal driver.

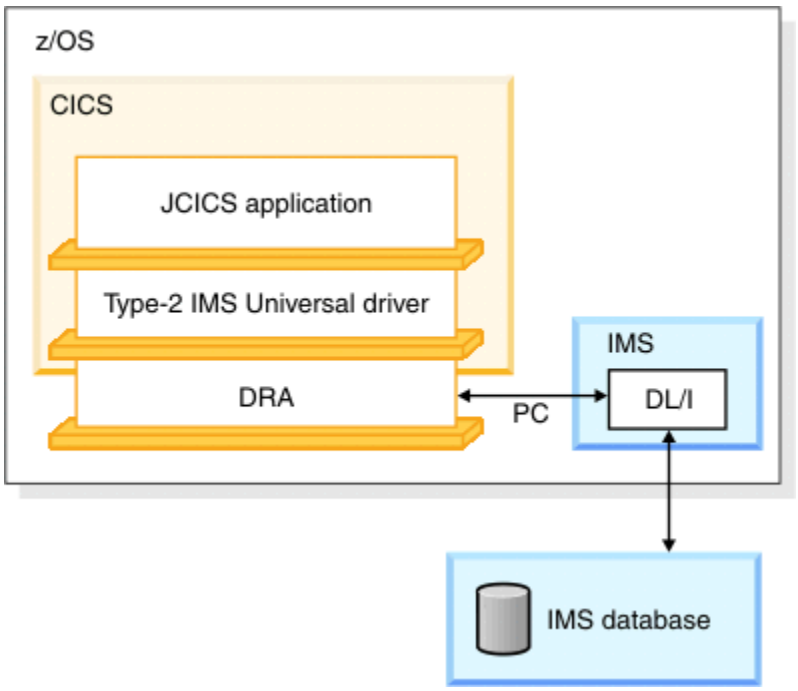


Figure 6. CICS application using a type-2 IMS Universal driver

Configuring CICS for the type-2 IMS Universal drivers

To run Java applications in a CICS environment that access IMS databases through a type-2 IMS Universal driver, you must install the type-2 IMS Universal driver in the IBM CICS Transaction Server for z/OS subsystem.

Prerequisite: Load the install files for the type-2 IMS Universal driver in a path that the CICS subsystem can access.

To configure CICS for a type-2 IMS Universal driver:

1. Build the IMS Universal driver OSGi bundle. To build the bundle:
   a) Write a bundle `Manifest.mf` file in a text editor. The following sample file is an example of `Manifest.mf`:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: IMS Universal driver OSGi
Bundle-SymbolicName: com.ibm.ims.osgi.Udb
Bundle-Version: 1.0.0
Bundle-ClassPath: imsudb.jar
Export-Package: com.ibm.ims.application,
com.ibm.ims.base,
com.ibm.ims.db,
com.ibm.ims.db.cci,
com.ibm.ims.db.hybrid,
com.ibm.ims.db.spi,
com.ibm.ims.dbd,
com.ibm.ims.dli,
com.ibm.ims.dli.conversion.util,
com.ibm.ims.dli.conversion.util.bidi,
com.ibm.ims.dli.converters,
com.ibm.ims.dli.dm,
com.ibm.ims.dli.logging,
com.ibm.ims.dli.t2,
com.ibm.ims.dli.tm,
com.ibm.ims.dli.types,
com.ibm.ims.dli.xa,
com.ibm.ims.drda.base,
com.ibm.ims.drda.converters,
com.ibm.ims.drda.db,
com.ibm.ims.drda.t4,
com.ibm.ims.drda.t4.util,
```

```
com.ibm.ims.drda.t4nativesql,
com.ibm.ims.jdbc,
com.ibm.ims.jdbc.batch,
com.ibm.ims.jdbc.xa,
com.ibm.ims.jms,
com.ibm.ims.opendb,
com.ibm.ims.psb,
com.ibm.ims.smf,
com.ibm.ims.xmldb,
com.ibm.ims.xmldb.dm,
com.ibm.ims.xmldb.shredder,
com.ibm.ims.xmldb.xms
Import-Package: com.ibm.cics.server;version="[1.300.0,2.0.0]";resolution:=optional
Bundle-RequiredExecutionEnvironment: JavaSE-1.7
```

b) Add both the `Manifest.mf` file and the IMS Universal driver `imsudb.jar` file to a zip archive.

c) Rename the zip archive to `com.ibm.ims.osgi.Udb_1.0.0.jar`.

d) Use the CICS explorer to deploy the created OSGi bundle.

2. Modify the CICS environment UNIX System Services file, DFHJVMPR, that contains the JVM profile.

a) Update the OSGI_BUNDLES variable so that it contains the path to the created OSGi bundle as follows:

```
OSGI_BUNDLES=pathprefix/com.ibm.ims.osgi.Udb_1.0.0.jar
```

b) Update the LIBPATH variable so that it contains the path to the `libT2DLI_64.so` file as follows:

```
LIBPATH_SUFFIX=pathprefix/usr/lpp/ims/ims15/imsjava/lib
```

3. Ensure that the CICS AIBTDLI is loaded over the IMS AIBTDLI interface.

a) Set the CICS SDFHLOAD member above the IMS SDFSRESL member in the CICS STEPLIB.

Related reading: For detailed information about CICS system definition, see the *CICS Transaction Server for z/OS CICS System Definition Guide*.

Running applications on CICS that use the type-2 IMS Universal drivers

To run a CICS application program that accesses IMS DB through a IMS Universal driver, you must perform several steps.

To run a Java application in CICS that accesses IMS DB through a type-2 IMS Universal driver, complete the following steps.

1. Start IMS DB and CICS.
2. Turn off the uppercase translation feature of CICS by entering: `CEOT  NOUCTRAN`
3. Define a program that can run the Java application (JVM class).
4. Define a transaction that can run the program.
5. Install the program that you defined to run the Java application (JVM class).
6. Install the transaction that you defined to run the Java application (JVM class).

Part 2. CPI Communications and APPC/IMS

These topics introduce CPI-Communications and APPC/IMS. The topics discuss how CPI-Communications driven application programs function and how to administer APPC/IMS and use APPC/IMS with the CPI Communications interface to build CPI application programs.

© Copyright IBM Corp. 1974, 2020

Chapter 2. CPI Communications

This topic introduces CPI Communications driven application programs and distributed Syncpoint protected conversations.

CPI-C driven application programs

A CPI Communications driven application program can use IMS-managed resources in two ways: by using the SQL calls to access Db2 for z/OS through the IMS External Subsystem (ESS) Attach Facility and by using the APSB call to allocate IMS resources.

If you use the SQL calls to access Db2 for z/OS through the IMS External Subsystem (ESS) Attach Facility and the Db2 for z/OS resource translation table (RTT) is not used, the initial Db2 for z/OS plan name is the application program name. After the APSB call, the Db2 for z/OS plan name is the PSB name specified in the APSB call.

You can use the following SAA resource recovery calls when you want an application program to commit or back out changes to IMS or Db2 for z/OS resources:

- Use the Commit call (SRRCMIT) to commit changes.
- Use the Backout call (SRRBACK) to back out changes.

SAA resource recovery commit processing

An application program tells IMS to commit changes to database resources by issuing the SAA resource recovery call, SRRCMIT.

By issuing the SRRCMIT call, an application program tells IMS to commit changes to database resources:

- Issue the SRRCMIT call when the application program updates any IMS resources or accesses Db2 for z/OS resources.
- Reissue the SRRCMIT call after making any subsequent changes to an IMS or Db2 for z/OS resource.
- Issue the SRRCMIT call before terminating your application program.

If the application program terminates with any uncommitted changes to IMS resources, IMS attempts to commit these changes. Dangling conversations are deallocated abnormally.

When you issue the SRRCMIT call, IMS gets control and generates an internal sync-point call (if the conversation was not allocated with SYNCLVL=SYNCPT). All database changes are committed. All messages inserted to alternate PCBs (program control blocks) are sent to their final destination.

Normal termination

In IMS, normal termination occurs when an application program terminates without abending. For a CPI Communications driven application program, an implicit commit occurs.

Definition: An *implicit commit* occurs when the application program does not issue an explicit call to commit the current transaction, but one of the following occurs:

- The application program terminates normally.

Backout processing

Transaction updates can be backed out for a variety of reasons.

Transaction updates are backed out when any of the following occurs:

- Backout is issued by the application program.
- The application program terminates abnormally.

© Copyright IBM Corp. 1974, 2020

- The application program terminates normally, but the implicit commit fails.
- IMS resources are inflight during IMS system restart.

The backout consists of the following actions:

- All database updates are backed out.
- All messages inserted to non-express alternate PCBs are discarded.
- All messages inserted to express PCBs that have not been enqueued are discarded.
- The APPC/MVS™ verb ATBCMTP TYPE=ABEND is issued. [1]

The application program tells IMS that a backout is required by issuing SRRBACK or by terminating abnormally.

Abnormal termination

When your application program abends, IMS backs out to the last IMS sync point.

IMS considers an application program to have terminated abnormally if either of the following occurs:

- The implicit commit fails.
- The application program abends.

Session failure

If any LU 6.2 session fails during the conversation, you can choose to end the application program or continue processing.

IMS TM is not involved and places no restrictions on your choice of committing or backing out updates. The application programmer makes this decision.

Because IMS TM is not informed of the session failure, it takes no action. The normal processing rules for commit and backout apply.

Return codes

Your application program receives return codes from IMS on the SAA resource recovery SRRCMIT and SRRBACK calls.

Your application program can receive the following return codes:

RR_OK
The backout or commit operation completed successfully. All protected resources if backed out have been returned to their previous sync point; if they have been committed, they have advanced to a new sync point, and all changes made during the logical unit of work have been made permanent.

RR_PROGRAM_STATE_CHECK
A non-CPI Communications driven IMS application program issued an SAA resource recovery Commit call. No commit or backout has been performed.

RR_BACKED_OUT
A resource manager voted "no" during sync point processing. The sync point was initiated by the SAA resource recovery Commit call. The resource state is backed out for all resources.

System restart/resolve-in-doubt processing

After a system failure, a key part of restart processing is known as *resolve-in-doubt* processing. If the system fails, IMS determines whether to perform resolve-in-doubt processing for IMS-protected resources.

Examples of IMS-protected resources are:

[1] Issuing the verb ATBCMTP causes all LU 6.2 conversations associated with this TPI to terminate with CM_DEALLOCATE_ABEND.

- IMS DB databases
- Db2 for z/OS databases
- IMS TM message-queue messages

If the IMS system fails before the transaction completes phase one of the two-phase commit process (sync point), IMS backs out during IMS restart. Backout includes transactions that were processing at the time of failure.

If the transaction completes phase one of the commit process, resolve-in-doubt processing can take place during IMS restart. If only IMS resources are affected, commit processing occurs. If Db2 for z/OS resources are involved, resolve-in-doubt processing occurs between IMS and Db2 for z/OS.

No transactions using explicit CPI Communications driven application programs are preserved across an IMS restart.

CPI-C application program recovery

No recovery processing exists for application programs using the explicit CPI Communications driven interface.

IMS discards all CPI Communications driven transactions at restart regardless of their state at the time of failure. Application program designers should be aware of the SAA resource recovery resynchronization functions and consider the impact on their application program designs.

The application program should provide full integrity by issuing a SAA resource recovery Commit or Backout call for session failures. Application programs that require recovery assistance must be standard DL/I application programs.

Related reference

CALL statement (Application Programming APIs)

Programming requirements

The calls that initiate implicit sync point (DL/I GU to the message queue, CHKP, and SYNC) are invalid for CPI Communications driven application programs, and receive status AD (function parameter invalid). The CPI Communications driven application program activates IMS sync point processing by issuing the SRRCMIT and SRRBACK calls.

If you allocate a conversation with `SYNCLVL=NONE` or `SYNCLVL=CONFIRM`, include module `DFSCPIRO` with your application program in the bind step. Including this module allows your application program to resolve the external references for SRRCMIT and SRRBACK.

No language-unique programming concerns exist in IMS for the SAA resource recovery interface.

Pseudonym files

APPC/IMS uses APPC/MVS services to provide SAA resource recovery support. APPC/MVS does not provide SAA resource recovery pseudonym files. However, you can create your own pseudonym files.

Related reading: For sample pseudonym files, see *SAA CPI Resource Recovery Reference*. These sample pseudonym files include examples on how to define working storage in the different languages.

Briefly, programmers of different languages need to define the following:

- IMS TM C programmers need to define z/OS as their operating system.
- IMS TM COBOL programmers must define their buffers in working storage.
- IMS TM FORTRAN programmers must define EXTERNAL statements for SRRCMIT and SRRBACK.

RRS and distributed syncpoint/protected conversations

Regardless of whether the SYNCLVL setting is NONE, CONFIRM, or SYNCPOINT, if RRS=Y, z/OS Resource Recovery Services is the sync point manager and coordinates the update and recovery of multiple

protected resources. RRS controls how and when protected resources are committed by coordinating with the resource managers, such as IMS, that have registered with RRS.

RRS supports the Common Programming Interface for Resource Recovery (CPI-RR), an element of the SAA CPI that specifies resource recovery and coordinates recovering local and distributed resources.

Definitions:

- A *protected resource* is a set of local or distributed data that is updated in a synchronized and controlled manner. In the APPC environment, a protected resource is a resource that is updated in an allocated conversation in which SYNCLVL=SYNCPT has been specified.
- A *resource manager* is a product, such as IMS, that owns protected data resources that are updated in an APPC conversational environment in which SYNCLVL=SYNCPT has been specified. IMS acts as a resource manager for DL/I data, Fast Path data, and the message queues.

The three participants in resource recovery include:

RRS (sync point manager)
Resource manager (such as IMS or Db2 for z/OS)
Application program

The following figure shows the three participants in the resource recovery process, and their interaction.

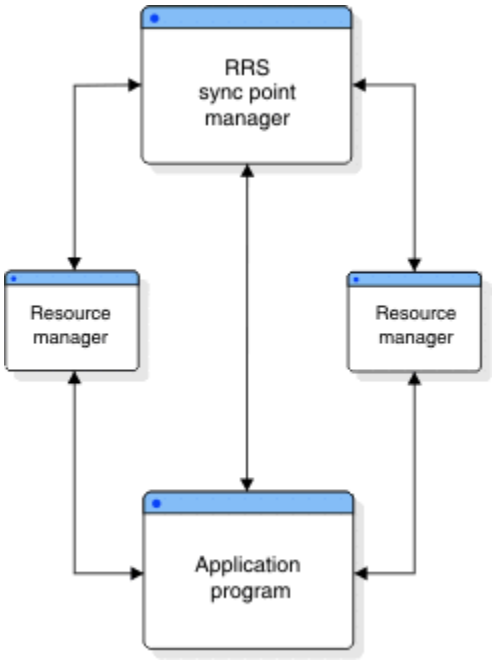


Figure 7. Participants in resource recovery

The two-phase commit protocol

The *two-phase commit protocol* is a process involving z/OS Resource Recovery Services (RRS) and the resource manager that ensures that updates made to a set of resources by an application program are either all made or none made.

The application program decides whether to commit its changes to the resources; this commit is made to RRS, which polls all of the resource managers as to the feasibility of the commit call. Each resource manager votes whether to commit the updates. This is called *phase 1* of the two-phase commit protocol.

After RRS has gathered the votes, *phase 2* begins. If all votes are to commit the updates, then the phase 2 action is to commit; otherwise, phase 2 results in a backout of the updates. System failures, communication failures, resource manager failures, or application program failures are not barriers to completing the two-phase commit protocol.

Definitions:

- A *unit of recovery* is a unit of work that spans one commit (synchronization) point to the next commit point.
- Units of recovery are termed *inflight* between the time they are created (or the previous sync point) until the resource manager votes to commit the updates. If the resource manager fails while units of recovery are inflight, the resource manager backs out all of the database updates on the subsequent start.
- Units of recovery are termed *indoubt* between the time when the resource manager votes to commit the updates and the time when the sync-point manager calls the resource manager to do the commit. If IMS fails while units of recovery are indoubt, IMS holds the database updates until they are resolved.

Local-resource recovery versus distributed-resource recovery

In a *local-resource recovery environment*, the recovery participants reside on the same z/OS system. In a *distributed-resource recovery environment*, the recovery participants and the updated resources are scattered across multiple systems.

In a distributed-resource recovery environment, the APPC/PC (APPC/protected conversation) resource manager is used to provide the communications for the sync-point calls to remote systems.

The following figure illustrates how a distributed recovery environment operates.

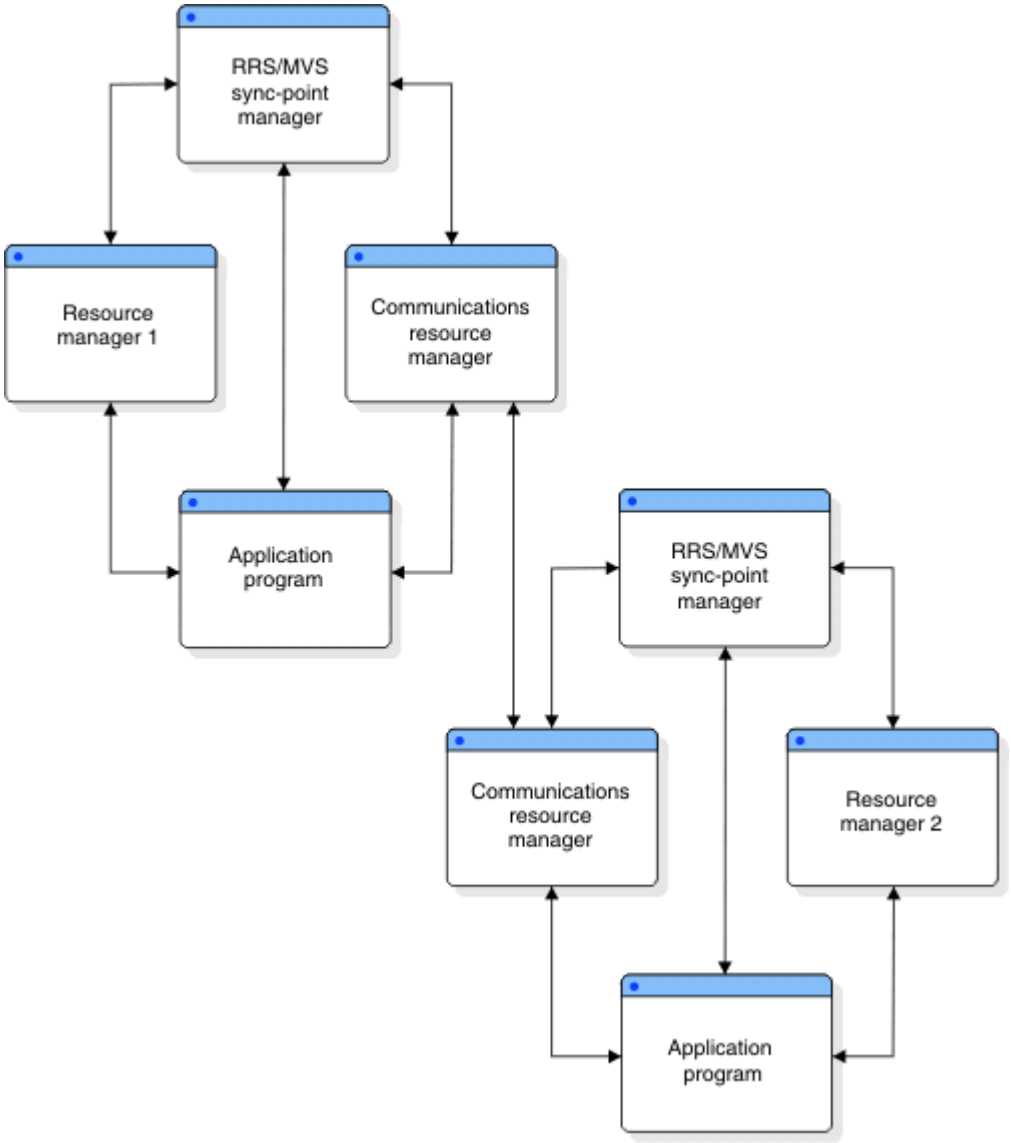


Figure 8. Distributed resource recovery

IMS as a resource manager

A resource manager controls a protected resource.

In general, a resource manager does the following:

- Provides an application programming interface (API) to allow application programs to access its resources
- Logs changes to data before making the changes permanent
- Logs unit of work status
- Participates in the commit or backout actions for updated resources
- Contains recovery mechanisms to restore data to a previous state

For its own resources, IMS is both the sync-point manager and the resource manager.

Within the two-phase commit protocol, IMS must do each of the following:

- Register with z/OS Resource Recovery Services (RRS) as a resource manager
- Participate in the sync-point process
- Express interest in the unit of work
- Recover its unit of work status after an outage

Registration

A component of RRS provides registration services so that IMS can identify itself as a resource manager. By registering, IMS is provided a set of services to aid in maintaining resource consistency associated with the protected conversation.

IMS registers each time a control region is started for a DB/DC active system on a z/OS system with the recovery platform support. In an XRF environment, the active system registers during its restart. The alternate system registers at the time of takeover.

Expressing interest

In addition to registering and supplying resource manager exit routines for specific stages of the two-phase commit protocol, IMS must also express interest in participating in the two-phase commit process for a particular unit of recovery.

Resolution of incomplete interests

In the event of an IMS or z/OS outage, during the IMS restart, the incomplete UR expressions of interest must be resolved.

RRS maintains unit of recovery information (such as identifier, state, and resource manager private data), which RRS presents to the restarting resource managers that previously expressed interest.

Sync-point participation

After IMS successfully registers and restarts, it supplies addresses of its exit routines to RRS. Several exit routines (such as prepare, commit, and backout) represent specific points in the two-phase commit protocol, which IMS can call to participate in the process.

Activating protected conversations

z/OS uses a construct with z/OS Resource Recovery Services (RRS) called a context.

Definition: A *context* is the entity for which resource managers perform services, to which they allocate resources and lock ownership, and in which they can express interest in participating in the protocol to ensure that the resource is updated in an orderly manner.

The type of context that the resource manager creates, owns, and manipulates is called the *private context*. A resource manager can create a context on behalf of another resource manager. RRS uses the private context to identify an application program's unit of work to maintain information for the resource manager concerning which of their resources are associated with the unit of work.

APPC as the communications manager

When APPC is the communications manager, RRS support is activated when a conversation is allocated with SYNCLVL=SYNCPT. This type of conversation is a protected conversation.

When SYNCLVL=SYNCPT is specified, APPC acquires a private context on behalf of IMS. IMS provides its resource manager name to APPC in its identity call. APPC provides the private context to IMS as the message header. IMS, using this context, then assumes the role of a participant in the two-phase commit process with the sync-point manager, RRS.

In addition to the SYNCLVL=SYNCPT, the keyword ATNLOSS=ALL must be specified in the VTAM definition file for whichever LUS need to be enabled for protected conversations.

Using OTMA with protected conversations

In an OTMA environment, OTMA is not a resource manager registered with RRS. The process remains an inter-process protocol between a server (IMS) and a number of clients (application programs). Therefore, OTMA cannot obtain a private context token to pass to IMS, as APPC does. The client-adapter code that uses OTMA is responsible for obtaining and owning a private context, and for providing the context ID. In messages passed between the partners, the context-ID field contains the context token (if it is a protected conversation).

When IMS finds the context-ID in the message, IMS assumes the role of a participant in the two-phase commit process, as it does in the APPC environment.

XRF and protected conversations

Running protected conversations (using RRS with either APPC/PC or OTMA) in an IMS-XRF environment does not guarantee that the alternate system can resume and resolve any unfinished work started by the active system. A failed resource manager must re-register with its original RRS system if the RRS system is still available when the resource manager restarts. Only if the RRS on the active system is not available can an XRF alternate system register with another RRS in the sysplex and obtain the incomplete unit of recovery data of the failing active system.

Recommendation: Because IMS retains indoubt units of recovery until they are resolved, switch back to the active system as soon as possible to obtain the unit of recovery information and to resolve and complete all the work of the resource managers.

Chapter 3. Administering APPC/IMS and LU 6.2 devices

This topic introduces APPC/IMS and describes how to administer APPC/IMS and LU 6.2 devices.

APPC/IMS overview

APPC/IMS, a part of IMS TM, lets you use the CPI Communications interface to build CPI application programs.

APPC/IMS allows distributed and cooperative processing between IMS and systems that have implemented APPC as shown in the following figure. APPC/IMS delivers support for APPC with facilities provided with APPC/MVS. (The APPC/IMS interface is provided by APPC/MVS and supports the CPI Communications interface. IMS TM supports the CPI resource recovery interface.) APPC/IMS supports the CPI resource recovery Commit (SRRCMIT) and Backout (SRRBACK) calls for IMS-managed local resources. These protected resources include:

IMS TM message-queue messages
IMS DB databases
Db2 for z/OS databases

APPC/IMS also supports the existing IMS DL/I application programming interface (API) enabling application programs to use LU 6.2 communications without the function of the CPI Communications interface. This allows most existing applications to continue to function with the APPC/IMS support of LU 6.2.

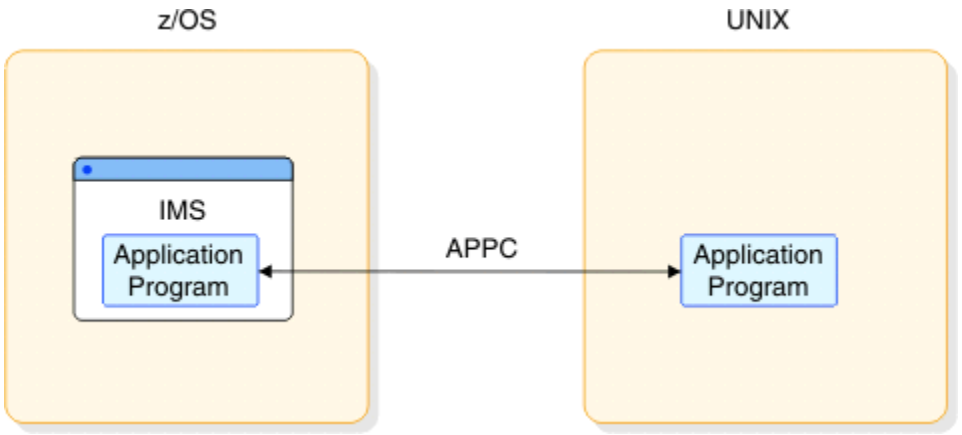


Figure 9. APPC support for IMS

Definitions:

- Within the context of administering IMS TM, "transaction programs," "applications, " "application programs," and "programs" are synonymous.
- Within the context of administering APPC/IMS and LU 6.2 devices, "APPC application programs" are synonymous with "LU 6.2 application programs."
- "LU 6.2 transactions" are those that originate from an LU 6.2 application program.

Recommendations: For APPC/IMS, do the following:

- Schedule your IMS standard or modified application programs entered from LU 6.2 remote systems using MSC. Be aware that CPI-C driven application programs cannot have transactions that execute on remote systems.
- Define your APPC/IMS LUs for use by APPC/MVS, as well as by any APPC application program.

© Copyright IBM Corp. 1974, 2020

- Use the LTERM and the MOD name in the first segment of your message. Use the LTERM to change the destination for your output to a non-LU 6.2 device. Use the MOD name to format error messages.
- Use a network-qualified LU name. You do not need to have unique names for the LUs on different systems.

IMS dependent regions are automatically defined to APPC as subordinate address spaces of the IMS Scheduler. An IMS BMP cannot be defined as an ASCH controlled application. It may use explicit conversation services through the IMS base LU.

IMS manages the APPC/IMS message buffers automatically; no definition is necessary. No special considerations are needed for EMH.

APPC/IMS flood control

The APPC/IMS flood control function helps prevent a sudden increase in the number of APPC/IMS transaction requests from exhausting IMS 31-bit private storage.

By default, APPC/IMS flood control is active and starts queuing incoming APPC transaction requests to 64-bit storage when the number of active APPC conversations reaches 5,000. If the flood condition worsens, APPC/IMS flood control stops all APPC input when the number of queued APPC requests in 64-bit storage reaches the default threshold of 10,000.

Except when all APPC input is stopped, APPC/IMS flood control does not apply to APPC requests that are used to submit IMS commands or to APPC requests that are received on back-end IMS systems in a shared-queues environment.

As the number of active and queued APPC requests nears these thresholds, IMS issues a warning message.

Modifying or disabling APPC/IMS flood control

You can modify or disable both the initial threshold for queuing requests to 64-bit storage and the secondary threshold that stops all APPC/IMS input by specifying the APPCMAXC=(*31_bit_max*,*64_bit_max*) parameter in the DFSDC*xxx* member in the IMS PROCLIB data set.

The *31_bit_max* value defines the maximum number of active APPC conversations that IMS can process concurrently before IMS starts queuing new APPC transaction requests to 64-bit storage. A specification of 0 completely disables APPC/IMS flood control.

You can view the current *31_bit_max* value by issuing the IMS type-1 command **/DISPLAY A DC**. If the displayed value is 0, APPC/IMS flood control is disabled.

The *64_bit_max* defines the maximum number of APPC transaction requests that can be queued in 64-bit storage before IMS stops all APPC input from z/OS. A specification of 0 disables the queuing of APPC requests in 64-bit storage.

You can clear all of the queued APPC requests in 64-bit storage by issuing the IMS type-1 command **/PURGE APPC**. The APPC conversation is rejected with the TP_Not_Available_No_Retry sense code.

When 64-bit queuing is disabled, if a flood condition occurs, the 31 bit maximum defines the threshold at which IMS stops all APPC input.

When APPC input is stopped

When IMS stops APPC input from z/OS, IMS does not itself reject incoming APPC requests, but rather issues a call to APPC/MVS to request that it stop sending any more APPC requests. Between the time that IMS issues the request and the time that APPC/MVS stops sending input, IMS can still receive APPC requests, so it is possible that the total number of APPC requests that are received or queued by APPC/IMS might exceed the defined maximum.

After APPC input is stopped, when the number of active APPC conversations in IMS 31-bit storage drops below 50% of the *31_bit_max* value, IMS automatically requests a resumption of APPC input from APPC/MVS.

IMS issues DFS4157E when APPC input is stopped.

VTAM alternative to APPC/IMS flood control

In addition to or as an alternative to the APPC/IMS flood control measures, you can specify a session limit for an individual logical unit (LU) in the VTAM ACB. VTAM stops sending messages to APPC/MVS after the session limit is reached. If only one LU is defined for an IMS system, the maximum number of active APPC requests is then the number of sessions that are defined in VTAM.

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

APPC/IMS application program interface

APPC/IMS has two distinct application program interfaces (APIs): the implicit and explicit interfaces. The same application program can use both APIs.

Implicit API

The *implicit API* is an extension of the IMS standard DL/I API (call xxxTDLI). It allows IMS application programs to communicate with LU 6.2 application programs without being sensitive to LU 6.2 protocols and without requiring the programmer to have any knowledge of LU 6.2. APPC/IMS provides functions not normally available to an LU 6.2 application program: message queuing, and automatic asynchronous message delivery and recovery. Existing IMS transactions use the implicit API to communicate with APPC.

Implicit API messages are placed on the IMS message queues or the Fast Path expedited message handling (EMH) buffers for Fast Path transactions. The originating IMS determines whether to mark the input messages as discardable or nondiscardable.

When the implicit API is used, IMS issues all required CPI Communications calls. The application program interacts strictly with the IMS message queues or the Fast Path EMH buffers.

Explicit API

The *explicit API* is the CPI Communications API and is available to any IMS application program. The application program makes calls to APPC using the CPI Communications interface without using IMS. These CPI calls are handled directly by APPC/MVS. Messages sent or received by the CPI Communications interface are not stored on the IMS message queues or in the EMH buffers, and these messages are not available for transaction restart. No IMS-provided functions are involved for these messages.

Alternatively, you can also use z/OS the ATB*xxxx* calls of the APPC/MVS TP services. For information on using these calls, see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*.

APPC/IMS application programs

APPC/IMS has three different types of application programs: standard, modified, and CPI Communications driven.

The application programs are defined as:

Standard
: No explicit use of CPI Communications facilities.

Modified
: Uses the I/O PCB to communicate with the original input terminal. Uses CPI Communications calls to allocate new conversations and to send and receive data.

CPI Communications driven
Uses CPI Communications calls to receive the incoming message and to send a reply on the same conversation. Uses the DL/I APSB call to allocate a PSB to access IMS databases and alternate PCBs.

You can schedule your standard and modified application programs locally and remotely using MSC. The logic for local application programs differs from the logic for remote application programs. In the following topics, the differences are described.

Standard IMS application programs

Standard IMS application programs use the existing IMS call interface.

Application programs that use the IMS standard API can take advantage of the LU 6.2 protocols. Standard IMS application programs use a DL/I GU call to trigger a sync point and to get the incoming transaction. These standard IMS application programs also use DL/I ISRT calls to generate output messages to the same or different terminals, which can be LU 6.2 terminals. (A non-message-driven BMP is considered a standard IMS application program when it does not use the explicit API.) The identical program can work correctly for both LU 6.2 and non-LU 6.2 terminal types. IMS generates the appropriate calls to APPC/MVS services.

IMS provides the following services for standard IMS application programs:

- Receives incoming transaction from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Schedules transactions into local and remote IMS dependent regions
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry
- Provides integration of IMS-controlled conversation flows with database updates during sync point for all APPC Sync_Level options (NONE, CONFIRM, SYNCPT)
- Provides all needed LU 6.2 calls to APPC/MVS services
- Sends either synchronous or asynchronous output to an LU 6.2 application program
- Keeps asynchronous output on IMS message queue until successful transmission
- Allocates new LU 6.2 conversations for messages inserted to alternate PCBs using the DL/I ISRT call

Existing application programs that are sensitive to a terminal's hardware characteristics, such as cursor position or MFS format names, might need to be changed to communicate with LU 6.2 application programs.

Restrictions:

1. If a LU 6.2 synchronous conversation implicit transaction initializes other transactions (program-to-program switch), an express PCB can not be used to do the ISRT. An express PBC causes race conditions to occur and the output may randomly return to the inputting terminal on a new asynchronous conversation with `TPNAME DFSASYNC`. The original conversation may not be deallocated.
2. If a transaction initializes more than one child transaction, which in turn may initialize other transactions, and one of the child transactions provides the response, the result is unpredictable.

   Depending on the execution sequence of these transactions the LU can receive a `DFS2082` message with the response sent to the default TP name `DFSASYNC` or the LU receives the response and no `DFS2082` message is issued.

MSC and standard IMS application programs

When an APPC application program enters an IMS transaction that executes on a remote IMS, an LU 6.2 conversation is established between the APPC application program and the local IMS.

The local IMS is considered the partner LU of the LU 6.2 conversation. The transaction is then queued on the remote transaction queue of the local IMS. From this point on, the transaction goes through normal

MSC processing. After the remote IMS executes the transaction, the output is returned to the local IMS, and is then delivered to the originating LU 6.2 application program.

The originating (local) IMS provides the following services:

- Receives incoming transaction from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Queues the transaction to its remote transaction queue
- Sends the transaction across the MSC link
- Receives the transaction response
- Sends either synchronous or asynchronous output to an LU 6.2 application program

The remote IMS provides the following services for the remote standard application program:

- Receives the incoming transaction from the partner IMS (originating or intermediate IMS) over the MSC link
- Schedules transactions into dependent regions
- Commits database changes at sync point
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry
- Keeps transaction output on the IMS message queue until the transmission is successful
- Returns the transaction output to the local IMS over the MSC link

Restriction: MSC is not supported if the originating LU 6.2 conversation is allocated with `SYNCLVL=SYNCPT`.

Modified IMS application programs

Modified IMS application programs use a DL/I GU call to retrieve the incoming transaction and to trigger a sync point.

These modified IMS application programs also use DL/I ISRT calls to generate output messages to the same or different terminals, which can be LU 6.2 terminals.[2] Unlike standard IMS application programs, modified IMS application programs use CPI Communications calls to allocate new conversations, and to send and receive data. IMS has no direct control of these CPI Communications conversations.

Modified IMS transactions are indistinguishable from standard IMS transactions until program execution. In fact, the same application program can be a "standard IMS" application on one execution, and a "modified IMS" application on a different execution. The distinction is simply whether the application program has used CPI Communications resources.

IMS provides the following services for modified IMS application programs:

- Receives incoming transactions from LU 6.2 application programs
- Schedules transactions into local and remote dependent IMS regions
- Appropriate transaction recoverability before transaction scheduling
- Provides integration of IMS-controlled conversation flows with database updates during sync point for APPC Sync_Level options (NONE, CONFIRM, SYNCPT)
- Provides all necessary LU 6.2 calls to APPC/MVS services for IMS-controlled LU 6.2 conversations
- Sends either synchronous or asynchronous output to LU 6.2 application programs
- Keeps asynchronous output on the IMS message queue until successful send occurs

[2] A non-message-driven BMP is considered a modified standard IMS application program when it uses the explicit API.

- Allocates new LU 6.2 conversations for any messages inserted to alternate PCBs using the DL/I ISRT calls

IMS does not provide any services for conversations explicitly allocated by the application program. Explicitly allocated conversations need to be deallocated if a program abend occurs.

MSC and modified IMS application programs

When an APPC program enters an IMS transaction that executes on an MSC remote IMS, an LU 6.2 conversation is established between the APPC program and the local IMS.

The local IMS is considered the partner LU of the LU 6.2 conversation. The transaction is then queued on the remote transaction queue of the local IMS. From this point on, the transaction goes through normal MSC processing. After the remote IMS executes the transaction, the output is returned to the local IMS, and then delivered to the originating LU 6.2 program.

The originating (local) IMS provides the following services:

- Receives incoming transaction from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Queues the transaction to its remote transaction queue
- Sends the transaction across the MSC link
- Receives the transaction response
- Sends either synchronous or asynchronous output to an LU 6.2 application program

The remote IMS provides the following services for the remote modified application program:

- Receives the incoming transaction from the partner IMS (originating or intermediate system) over the MSC link
- Schedules transactions into dependent regions
- Appropriate transaction recoverability before transaction scheduling
- Commits database changes at sync point
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry
- Keeps transaction output on the IMS message queue until the transmission is successful
- Returns the transaction output to the local IMS over the MSC link

Restriction: MSC is not supported if the originating LU 6.2 conversation is allocated with `SYNCLVL=SYNCPT`.

CPI Communications driven application programs

CPI Communications driven application programs are defined only in the APPC/MVS TP_Profile data set; they are not defined to IMS.

The CPI Communications driven application program definition is dynamically built by IMS when a transaction is presented for scheduling by APPC/MVS based on the APPC/MVS TP_Profile definition after IMS restart. The definition is keyed by TP name. APPC/MVS manages the TP_Profile information.

When a CPI Communications driven transaction program requests a PSB, the PSB must already be defined to IMS by using the APPLCTN macro during system definition and by generating the appropriate PSBs and ACBs with the Program Specification Block (PSB) generation utility and the Application Control Blocks Maintenance utility when APPLCTN PSB= is specified. When APPLCTN GPSB= is specified, generating PSBs and ACBs is not required.

CPI Communications driven application programs must use CPI Communications calls to accept the incoming conversation and to send a reply on the same conversation. The DL/I GU call is not used to

retrieve the initiating transaction from the LU 6.2 application program. No IMS resources are allocated when the application program is scheduled. Instead, the application program can use the DL/I APSB call to allocate a PSB that provides access to IMS databases and to alternate PCBs. A CPI Communications driven application program can send messages to other terminals (either LU 6.2 or non-LU 6.2) or other IMS transactions (either local or remote) by inserting to an alternate PCB, after allocating the appropriate PSB. Both the explicit and implicit API can be used on the same application program.

IMS provides the following services for CPI Communications driven application programs:

- Schedules the transaction.

  IMS does not receive input before scheduling. It does not interact with the conversation at any time other than to possibly reject the inbound allocate request. If IMS rejects the inbound allocate request, the transaction is not scheduled.
- Provides sync point of local resources.
- Schedules PSB if called by application program.
- Processes calls to alternate or database PCB made by the application program.

Related concepts

"RRS and distributed syncpoint/protected conversations" on page 27
Regardless of whether the SYNCLVL setting is NONE, CONFIRM, or SYNCPOINT, if RRS=Y, z/OS Resource Recovery Services is the sync point manager and coordinates the update and recovery of multiple protected resources. RRS controls how and when protected resources are committed by coordinating with the resource managers, such as IMS, that have registered with RRS.

"CPI Communications" on page 25
This topic introduces CPI Communications driven application programs and distributed Syncpoint protected conversations.

Designing an application for APPC (Application Programming)

Using the MOD name and LTERM interface

Your LU 6.2 application program can use an interface to emulate MFS.

For example, the application program can use the MOD name to communicate with IMS to specify how an error message could be formatted. For non-LU 6.2 application programs, the MOD name is given to the MFS formatting modules in IMS; for LU 6.2 application programs, the MFS modules are not called, and the MOD name is given to the LU 6.2 Edit exit routine (DFSLUEE0) as a parameter. The LU 6.2 Edit exit routine can do whatever the programmer specifies with the MOD name, such as format an error message.

Your LU 6.2 application program uses the LU name to send data to an LU 6.2 application program. However, if you want to send data to a non-LU 6.2 device such as a printer, you can use the LTERM instead of the LU name.

The Initialization exit routine (DFSINTX0) can be used to create a user table of MOD names that you might want to use for formatting messages, and LTERMs that you might want to use as printers. This user table can be used by DFSLUEE0 to find the appropriate MOD name or LTERM for your application program.

LU 6.2 application programs can send both the LTERM and the MOD name in the first segment of the message. The LU 6.2 Edit exit routine (DFSLUEE0) checks the contents of the first message segment. Based on the information it finds in a user table, the exit routine decides whether to return the LTERM and the MOD name to IMS. IMS saves the LTERM and the MOD name in the I/O PCB. For formatting output, IMS provides the address of the MOD name in the first segment of the message to the LU 6.2 Edit exit routine (DFSLUEE0). For changing the destination to a non-LU 6.2 device, IMS provides the LTERM in the first segment of the message to the LU 6.2 Edit exit routine (DFSLUEE0). The Initialization exit routine (DFSINTX0) can be used to create the user table. This exit routine must pass the address of the user table to IMS, and IMS passes the address to DFSLUEE0.

Establishing APPC/IMS

Before activating APPC/IMS, an IMS system definition is needed to specify 390 as the third parameter of the SYSTEM keyword for the IMSCTRL macro.

CPI Communications driven application programs and LU 6.2 application programs cannot be defined in a system definition. LU 6.2 application programs are only defined to VTAM.

Start APPC/IMS by specifying APPC=Y on the IMS startup parameter. The default is APPC=N. When 'N' is specified, a connection to APPC/MVS services is not established during IMS initialization. When 'Y' is specified, IMS establishes a connection with APPC/MVS during IMS initialization. The /START APPC command overrides APPC=N.

TP_Profile

The TP_Profile is a VSAM data set owned by APPC/MVS and maintained by the APPC/MVS Administration utility (ATBSDFMU) or by the administrator using TSO/ISPF dialogs. The purpose of the TP_Profile entries is to provide attribute information for TP names.

CPI Communications driven application programs must be defined in the APPC/MVS TP_Profile. IMS system-defined transaction codes can optionally be defined in a TP_Profile. The existence of an IMS definition (in the IMS GEN or by online change) causes the transaction to be considered a standard DL/I or modified-standard application.

The TP_Profile, an APPC/MVS resource, contains definitions of transaction program names (TPNs) and their characteristics. You can define a TP_Profile to schedule an IMS transaction program that uses a transaction code that is different from the TPN.

IMS uses the TP_Profile to establish transaction scheduling characteristics for CPI Communications driven application programs. Based on the IMS dependent section of the APPC/MVS TP_Profile definition, IMS dynamically defines these characteristics when a transaction is presented for scheduling after restart by APPC/MVS.

CPI Communications driven transaction programs are defined only in the TP_Profile. The definition is by TPN. The same TPN can be defined differently for different LU names by using a different TP_Profile data set. The LU name is associated with an IMS.

The default TP_Profile data set name is SYS1.APPCTP. The LUADD TPDATA option in the SYS1.PARMLIB (APPCPMxx) member specifies the TP_Profile data set name used for this LU.

Use TP_Profile dialog or the APPC/MVS administration utility (ATBSDFMU) to define a TP_Profile.

Example: The following figure is an example of the IMS-specific area of the TP_Profile definition (Panel 1).

```
 ----------------- IMS TP_Profile Panel -----------------

TP Name . . . : INQUIRY_Part

Transaction Code . . . . . PART
Security Type  . . . . . _____     (NONE,CHECK,FULL, default=CHECK)

CPI Communications Driven Options
   Transaction Class . . . ___       (Range 1 - 999, default=1)
   Maximum Regions . . . . ___       (Range 0 - 999, default=1)

Comments . . .       (Optional 1 to 10 lines)
> TP_PROFILE Created 10/8/91_____________________________________________ <
> Access IMS Sample Parts DATABASE via. program DFSSAM02_________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <
> _____________________________________________________________________ <

 PF01 = Help     PF03 = Exit     PF12 = Cancel     Enter = Accept
```

Figure 10. IMS-specific TP_Profile panel 1

To maintain IMS TP_Profiles using ISPF, do each of the following:

1. Enter TSO ICQASRM0 from the TSO command line of the TSO/E to start the ISPF TP_Profile System Data Facility Maintenance Utility from a TSO user ID. If this utility is not available, contact your z/OS system programmer.
2. Enter **S** next to the TP_Profile selection and the TP_Profile data set name specified on the TPDATA keyword on the LUADD statement for your IMS LU. (The LUADD statement is in the APPCTPxx PARMLIB member, where xx is the APPC suffix.)
3. A list of TP_Profiles is displayed. Select **A** to add a new TP_Profile or **E** to edit an existing TP_Profile. If you are adding a TP_Profile, you must supply a scheduler name. This name was set at IMS installation time. The recommended name is IMS .
4. After the general TP_Profile characteristics are supplied, the ISPF editor panel is displayed. Enter DFSTPROF on the command line to display the IMS TP_Profile Maintenance panel.
5. Supply IMS scheduler-dependent characteristics. Press enter to save your changes or PF3 or PF12 to cancel your changes. You can press PF1 for online help on fields supplied in this panel.

The TP_Profile name is not available on all releases of TSO/E, so a value of "Not Available" is displayed. This does not suggest that a problem exists.

Related reading: For more information about this utility, see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*.

APPC/MVS Administration utility (ATBSDFMU) example

The following example is an APPC/MVS Administration utility (ATBSDFMU) entry.

```
TPADD TPSCHED_EXIT(DFSTPPE0)
      TPNAME(INQUIRY_PART)
      SYSTEM
      ACTIVE(YES)
      TPSCHED_DELIMITER(##)
          TRANCODE=PART
          CLASS=1
          MAXRGN=1
                                CPUTIME=0
          ##
```

In this example, the IMS section starts with TRANCODE=PART. The other control statements are shown for completeness.

The IMS TP_Profile parsing module, DFSTPPE0, performs validity checking and parses the input data in IMS. This module should be loaded into the STEPLIB data set of the step that adds the TP_Profile. The APPC/MVS Administration utility (ATBSDFMU) requires that STEPLIB be APF authorized.

The following five keywords are used to add an IMS section to the TP_Profile entry. The keyword-parameter sets must be separated by one or more blanks. The keyword-parameter sets must be specified between columns 1-72. An asterisk (*) in column 1 indicates a comment.

TRANCODE= 1 - 8 characters
Name of the IMS transaction code associated with this TP name consisting of alphanumeric or '#', '$', '@'. IMS translates the TP name to the TRANCODE. IMS scans for valid characters (00640 character set). If invalid characters exist, IMS uses the default transaction code, IMSTRAN, instead of a transaction code with the non-00640 characters.

CLASS= 1 - 999
Specifies the class used for scheduling. The default value is 1.

Recommendation: Define CPI transactions with a different message class from that used for non-CPI transactions. IMS handles all CPI transactions as priority zero within the transaction class.

MAXRGN= 0 - 999
Restricts the number of dependent regions that this CPI Communications driven transaction program can use. The default value is 1.

RACF®=NONE, CHECK, or FULL
RACF=NONE causes IMS to call the Transaction Authorization exit routine (DFSCTRN0).

RACF=CHECK causes IMS to call RACF for security checking when IMS receives a transaction (using RCLASS of TIMS or CIMS), but does not clone the security environment into the dependent region when the transaction executes.

RACF=FULL clones the security environment to the dependent region at execution time. Specifying this parameter and issuing the IMS command /SECURE APPC PROFILE enables APSB SAF Security for this CPI-C application program.

CPUTIME= 0 - 1440
Specifies the number of CPU seconds that the CPI-C program is allowed to use. If it exceeds the limit, it is terminated with ABENDU0240. This time limit protects against program loops, which locks resources from other applications. The default is 0, which is no limit.

You can use the TP_Profile entry in two ways:

- To specify an IMS transaction code that is defined in the IMS. The CLASS and MAXRGN parameters in the TP_Profile are ignored and the transaction values in IMS remain unchanged. The TP_Profile entry provides mapping for a 64-character TPN into an 8-character transaction code.
- To specify an IMS transaction code that is not defined in the IMS. The IMS transaction code is a CPI Communications driven transaction, and is used as the load module name of the scheduled application program and the dynamically built transaction name.

When a TP_Profile is not defined, IMS uses the first 8 bytes of the TPN translated to the IMS character set as the transaction code.

The allocate request is rejected if the transaction code is not valid.

Related reading: For more information about using the APPC/MVS Administration utility (ATBSDFMU), see z/OS MVS Planning: APPC/MVS Management.

Outbound LU specification

You can specify a defined APPC LU as the outbound LU.

The default setting for defined APPC LUs is BASE LU. Changing an outbound LU is useful because, when the outbound LU has status of disabled, IMS does not allocate the APPC conversation.

The outbound LU must be defined in the APPCPMxx member of the SYS1.PARMLIB library. To specify an LU as the outbound LU, use the OUTBND= parameter in the DFSDCxxx PROCLIB member. You can set the

outbound LU by using the /CHANGE APPC OUTBND command. However, a restart sets the outbound LU to the value in the DFSDCxxx member, if specified. If it is not specified, the outbound LU is set to BASE LU.

Outbound side information

APPC/MVS *side information* supplies destination information, such as the name of the partner program, the name of the LU at the partner node, and the logon mode name.

CPI Communications provides a way to use system-defined values for these required fields; these system-defined values are the side information. This information can be used by an IMS application program allocating (establishing) an APPC conversation using CPI Communications, an IMS LU 6.2 descriptor, a DL/I change call (CHNG), or a DFSAPPC message switch.

System programmers supply and maintain the side information for CPI Communications programs.

Side information is accessed by a *symbolic destination name*. The symbolic destination name, called *sym_dest_name* within the context of administering IMS TM, corresponds to an entry in the side information file containing the following information:

partner LU name
Shows the name of the LU where the partner program is located. This LU name is any name for the remote LU that is recognized by the local LU for allocating a conversation. An example is a USERVAR name.

This LU name can be a 17-byte network-qualified LU name.

logon mode name
Used by LU 6.2 to designate the properties for the session that will be allocated for the conversation. The properties include the class of service to be used on the conversation. The network administrator defines a set of mode names used by the local LU to establish sessions with its partners. The system programmer uses one of these values in a side table entry. An invalid mode name prevents a conversation from being allocated.

TP name
Transaction program (TP) name specifies the name of the remote application program.

IMS and z/OS do not accept blank *sym_dest_name* values on the Initialize_Conversation call.

The default name for the side information file is SYS1.APPCSI. Define this file in the SYS1.PARMLIB(APPCPMxx) as shown in the following example.

```
SIDEINFO
     DATASET(SYS1.APPCSI)
```

The destination name, partner LU name, mode name, and TP name can be defined using the APPC/MVS Administration utility (ATBSDFMU) as shown in the following example.

```
SIADD
     DESTNAME(DESTX)
     TPNAME(LU62USER_TPX)
     MODENAME(APPCMODE)
     PARTNER_LU(APPCLUX)
```

Related reading: For more information on APPC calls, see *CPI Communications Specification*.

PARMLIB member

The APPC address space uses the APPCPMxx member of SYS1.PARMLIB. Define IMS as a local APPC component LU that is controlled by the APPC address space.

The scheduler name is the same IMSID used in the IMSCTRL macro. When IMS identifies to APPC, it passes its IMSID as the scheduler name SCHED(IMS1) in APPC member APPCPMxxx. The following is an example of the APPCPMxx member:

```
LUADD
     ACBNAME(IMSLU62)
```

```
     SCHED(IMS1)
     BASE
     TPDATA(SYS1.APPCTP)
     TPLEVEL(SYSTEM)
```

For XRF add:

```
USERVAR=uservar_name ALTLU=luname
```

The LUADD option keywords are defined below:

ACBNAME=*local LUNAME of IMS*

SCHED=*IMS id*

BASE*mandatory parameter*

TPDATA(*TP_Profile dataset name*)

TPLEVEL(*system*) suggested value

USERVAR=(*uservar_name*)

ALTLU=(*LUNAME*)

Related reading: For more information on these keywords, see z/OS MVS Planning: APPC/MVS Management.

Communication between VTAM and its application programs requires an ACB (application control block) whose name must be identically defined in the SYS1.VTAMLST APPL statement and in the APPCPMxx LUADD statement ACBNAME parameter.

APPC manages the IMS ACB. When IMS identifies to APPC, APPC gives IMS the name of the APPC-managed ACB name (LUNAME). The APPC LUNAME is not defined in IMS, because IMS does not manage the ACB. The entries in the SYS1.PARMLIB member APPCPMxx include both the IMS scheduler name (IMSID) and the LUNAME ACBNAME(xxxxxxx) that ties an IMS to an LUNAME.

This ACBNAME must be different from the ACBNAME used by IMS for non-LU 6.2 terminals. APPC/MVS expects its LUs to be defined as VTAM resources so that these LUs can access the SNA network. A VTAM application program (APPL) definition macro must be coded for each APPC/MVS LU. LU 6.2 application programs are only defined to VTAM, not to IMS. The SYS1.VTAMLST member example follows:

```
IMSLU62 APPL ACBNAME=IMSLU62
             APPC=Y
             …
```

APPC/MVS Timeout Service

Using APPC/MVS Timeout Service, you can indicate the maximum time interval an application waits before terminating a conversation and regaining control from APPC/MVS callable services.

When APPC/MVS does not respond to an APPC call, due to a network delay for instance, the dependent region hangs and the caller cannot regain control.

The timeout feature is activated at startup by specifying the APPCIOT=(*mmmm*:*ss*,*mmmm*) parameter in the DFSDCxxx member of the IMS.PROCLIB data set. The APPC time-out values are specified in minutes (*mmmm*) and seconds (*ss*). Valid values for *mmmm* are 00 to 1440. Valid values for *ss* are 00 to 59. If APPCIOT=00, there is no time-out detection. When a transaction is terminated due to timeout, messages DFS1965E and DFS1959E are sent to the MTO terminal and the z/OS console. The timeout value can be changed using the **/CHANGE** command.

For synchronous APPC conversations, if APPC timeout is active, then IMS uses ATBSTO6 service (SET_TIMEOUT_VALUE) to set the timeout value for each conversation.

For asynchronous APPC conversations, if APPC timeout is active, then IMS sets the timeout value when the conversation gets allocated. In either case, the timeout value is active until the conversation is deallocated, which occurs, in the case of IMS conversational transactions, when the IMS conversation ends.

Common Programming Interface Communications (CPI-C) transactions are not automatically supported by APPC/MVS Timeout service, but can exploit APPC/MVS Timeout service using ATBSTO5 service provided the proper coding is supplied.

Related Reading:

- For more information on programming MVS services, see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*.

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

APPC/MVS Error Extract Service

Whenever an APPC/MVS service call returns an unexpected return code, IMS issues APPC/MVS Error Extract Service call ATBEES3 with a DFS1995E prefix.

Related reading: For more information on ATB return codes, see:

- *z/OS MVS System Messages, Vol 3 (ASB-BPX)*
- *z/OS MVS Dump Output Messages*

Initializing and changing LU 6.2 descriptors

LU 6.2 descriptors allow the system programmer to specify an LTERM that associates an output destination with an LU 6.2 application program. This allows the system programmer to change the application program's destination using alternate PCBs to LU 6.2 application programs, without requiring any application program coding changes.

LU 6.2 descriptors are optional, but they are required if you want to dynamically create queue control blocks and define processing options.

The application program uses an LTERM name as a symbolic destination; only the system programmer needs to be aware of the actual term associated with this name.

The LU 6.2 descriptor entry contains:

- APPC/MVS side information entry name; this parameter can be omitted
- APPC conversation type (BASIC or MAPPED)
- APPC Sync_Level options (NONE, CONFIRM)
- LTERM name
- LU name of the destination of the LU 6.2 application program (overrides side information); this can be a network-qualified LU name up to 17 bytes in length
- The name of the local LU that IMS uses to allocate the outbound conversation, specified in the OUTBND parameter
- TP name to be scheduled (overrides side information)
- VTAM mode table entry to be used (overrides side information)

Here's an example of an LU 6.2 descriptor:

```
U L62TERM1 LUNAME=L62IMS1 TPNAME=CPICTRN1 MODE=L62MDE02
U L62TERM1 SYNCLEVEL=N OUTBND=MYLU02
```

Do not code a parameter and leave it blank (such as SIDE=ƀ), or an error message is issued. Instead, omit the parameter completely.

These LU 6.2 descriptor LTERMs are only for output and are never used by IMS as an LTERM name associated with an input message. DFSAPPC is an IMS-reserved name for the message switch facility.

The LU 6.2 descriptors are built as specified in IMS PROCLIB member DFS62DTx during IMS initialization. They can be added, deleted, or changed without restarting IMS. You can specify any number of descriptors. If an error occurs, the z/OS system console and the IMS JOBLIB record the error messages. IMS initialization continues regardless of any errors during descriptor initialization.

To add descriptors while IMS is running, you must first define the LU 6.2 descriptors in PROCLIB member DFS62DTx. Load the LU 6.2 descriptor from the IMS PROCLIB using the /START DESC command. To delete descriptors, use the /DELETE DESC command. To change descriptors, use the /CHANGE DESC command.

Related reading: For more information on coding these parameters, see *IMS Version 15 System Definition*.

Using MSC in an APPC/IMS environment

APPC/IMS uses the services of APPC/MVS and MSC to provide the communication interface for an MSC configuration.

Together, MSC and APPC/IMS allow:

- LU 6.2 programs to use the TP name of an IMS remote standard application program or an IMS remote modified application program. (The transaction is sent to the remote IMS and executes. The transaction's reply is sent across the MSC links to the local IMS and then on to the LU 6.2 application program.)
- A message switch to a remote logical terminal (LTERM) through the DFSAPPC System Service.
- Use of DFSAPPC for sending IMS remote transactions and data.
- Immediate or deferred program-to-program switching to an MSC-routed remote application program.

CPI Communications driven application programs cannot include transactions that execute on remote IMS systems.

All IMS transaction types except Fast Path are supported: conversational, nonconversational, response mode, and nonresponse mode.

IMS adds a prefix to the LU 6.2 message when the message is sent over an MSC link. The minimum size of this prefix is 480 bytes. The buffer sizes defined for MSC links should be large enough to hold at least one complete message. Valid MSC buffer sizes are 1024 bytes to 65536 bytes.

To change the input message destination to any IMS local or remote destination after a message is received but before it is processed, use the TM and MSC Message Routing and Control user exit (DFSMSCE0).

Definitions: Using MSC with APPC/IMS requires you to understand the terminology used for the different MSC systems:

- The *originating system (local)* is the system from which the LU 6.2 program enters the IMS transaction.
- The *remote system* is the system in which the remote transaction executes.
- The *intermediate system* is the IMS that routes messages between the local and remote systems.

At any time, any of these three systems can receive LU 6.2 transactions.

Related concepts

“Overview of Multiple Systems Coupling” on page 667

Multiple Systems Coupling (MSC) makes it possible for transactions to be entered in one IMS and processed in another IMS.

Recovering APPC transactions in an MSC environment

The recoverability of an IMS LU 6.2 transaction depends on whether the message is recoverable, irrecoverable, discardable, or nondiscardable, and when an error occurs.

You can determine the recoverability of APPC messages in an MSC environment. Resource failures affect recovery.

To recover APPC transactions in an MSC environment, analyze the types of failures that can occur. How you handle the error depends on the following:

- The resource that fails: Was it an LU 6.2 session failure, an IMS failure, an application program failure, or an MSC link failure?
- Transaction mode: Was it recoverable or irrecoverable?
- Transaction type: Was it local or remote?
- LU 6.2 conversation mode: Was it asynchronous or synchronous?

You are in control of recovery by the way you define the transaction. The information in the following topics highlights pertinent facts, and then points you to other areas in the IMS library where the subjects are explained in greater depth.

Recoverable versus nonrecoverable transactions

By coding the INQUIRY= keyword on the TRANSACT macro, you tell IMS the recovery status of a transaction. Non-inquiry mode transactions are recoverable; inquiry-mode transactions are not recoverable unless the RECOVER parameter is specified on the TRANSACT macro.

Recoverable transactions are recovered across any IMS failure, shutdown, or restart unless a COLDSTART, COLDSYS, or COLDCOMM restart is performed.

You must define remote transactions with identical recoverability attributes on the local system where the LU 6.2 session originates and on the remote system where the transaction is processed by the application program. You do not need to define the transaction on any intermediate IMS.

Message switches (messages from one LTERM to another) are always recoverable.

Related concepts

Recovery considerations for multiple systems (Operations and Automation)

Local APPC transaction discardability versus nondiscardability

The LU 6.2 protocol that you choose for sending a transaction to IMS and the transaction mode (recoverable or irrecoverable) you choose determine if a local APPC transaction is discardable or nondiscardable.

IMS discards a local APPC transaction when it is any of the following:

- A CPI Communications driven application program (without SYNCLVL=SYNCPT specified)
- It is defined as inquiry-only and nonrecoverable
- It is the result of synchronous input from the LU 6.2 application program
- It uses the APPC Sync_Level option NONE

Otherwise, the transaction is nondiscardable. IMS recovers nondiscardable transactions whenever possible; it never recovers discardable transactions.

Transaction processing point of failure

The point of failure in the processing of a transaction also affects its recoverability.

For example, when a local or remote transaction processes and reaches a commit point (sync point), IMS recovers the output response (from the log) even if you have defined the transaction as irrecoverable. Local APPC discardable transactions reach a commit point after IMS sends the output response message to the inputting APPC application program. In this situation, IMS has no output response message to recover or discard if a failure occurs after the commit point. If IMS has queued the transaction on an MSC link, IMS recovers the transaction across link failures.

A message can be either recoverable or irrecoverable, and either discardable or nondiscardable, according to the type of failure that might occur. The descriptions in this topic show you what happens to your transaction when the LU 6.2 session, MSC link, local IMS, intermediate IMS, remote IMS, or application program fail. This information assumes that you can recognize where a failure has occurred and what you need to do to recover.

Recovering transactions after an LU 6.2 session failure

If an LU 6.2 session fails while IMS is receiving an input message, IMS discards the message.

If IMS receives the complete message, processing depends on whether the conversation is:

CPI-C or not CPI-C
Synchronous or asynchronous
Local or remote

CPI-C transaction

If an LU 6.2 session fails while processing a CPI-C transaction, the application program can choose to end the conversation or to continue processing. IMS TM is not involved, and places no restrictions on the choice of committing or backing out updates. The application program makes the decision. Because IMS TM does not know about the session failure, it takes no action. The normal processing rules for commit and backout apply. IMS does not recover the LU 6.2 conversation.

Related reading: For information on designing your CPI-C LU 6.2 application program, see *IMS Version 15 Application Programming*.

Not a CPI-C transaction

If an LU 6.2 session fails while a local IMS is sending transaction output that is not CPI-C to the LU 6.2 program, and the conversation is synchronous, IMS calls the Message Control/Error exit routine to determine whether to abort and back out, or to continue processing. The default action is to stop the transaction and discard the output message (this is the mode of operation for all protected conversations; that is, conversations allocated using SYNCLVL=SYNCPT). If the conversation is asynchronous, IMS does not call the Message Control/Error exit routine, but queues the output on the message queue to the TP name of DFSASYNC.

Related reading: For information on coding the Message Control/Error exit routine, see *IMS Version 15 Exit Routines*.

Remote APPC transaction

If an LU 6.2 session fails while processing a remote APPC transaction, IMS recovers the output message if it has been enqueued on the local system's MSC link. If the transaction has not at least reached the point of being enqueued on the MSC link, IMS discards it. IMS discards the transaction regardless of the recoverability mode and regardless of whether the LU 6.2 conversation is synchronous or asynchronous. IMS does not call the Message Control/Error exit routine:

- If the transaction is asynchronous, when the output from a remote transaction for a failed LU 6.2 session returns from the remote system to the originating system, IMS sends the response asynchronously to the LU 6.2 application program by using the DFSASYNC TP name.
- If the transaction is synchronous, when the output from a remote transaction for a failed LU 6.2 session returns from the remote system to the originating system, IMS calls the Message Control/Error exit routine to either discard or re-route the transaction output. The default action is to discard the output.

Related reading: For information on using DFSASYNC in your application program, see *IMS Version 15 Application Programming*.

Related concepts

"CPI Communications" on page 25
This topic introduces CPI Communications driven application programs and distributed Syncpoint protected conversations.

Recovering transactions after an MSC link failure

When an MSC link failure occurs, IMS always recovers all messages, including IMS transactions and responses that are enqueued, about to be enqueued, or being sent across an MSC link. This recoverability is guaranteed, regardless of whether the message is en route to a local, intermediate, or remote MSC system.

The recoverability is not affected by the transaction mode (recoverable or irrecoverable) or the discardable or nondiscardable characteristics of the LU 6.2 protocol used to send the transaction to the local IMS.

Link failures can delay messages from other IMS systems and cause the synchronous LU 6.2 conversation to wait longer than expected for the response.

Related tasks

Restarting a logical link (Operations and Automation)

Recovering transactions after a local IMS failure

IMS discards local APPC transactions across a local IMS failure if they meet the discardability criteria. Otherwise, IMS does not discard local APPC transactions, because they are nondiscardable.

If you define your remote APPC transactions as inquiry-type transactions and do not specify RECOVER on the TRANSACT macro definition in the local IMS, IMS does not recover them after a local IMS failure. Otherwise, IMS recovers all recoverable, nonconversational transactions.

After the local IMS sends the transaction message to an intermediate or remote IMS, a local IMS failure has no affect on your transaction's recoverability. The transaction continues to its destination and is processed. When the remote IMS sends the transaction response to the originating IMS after the failure, IMS sends the response to its destination asynchronously through the default transaction program name (TPN) DFSASYNC. The LU 6.2 application programmer needs to plan for this situation.

Related concepts

"Local APPC transaction discardability versus nondiscardability" on page 47
The LU 6.2 protocol that you choose for sending a transaction to IMS and the transaction mode (recoverable or irrecoverable) you choose determine if a local APPC transaction is discardable or nondiscardable.

Designing an application for APPC (Application Programming)

Recovering transactions after a remote IMS failure

When a remote IMS failure occurs, based on the recoverability attributes of an APPC transaction in a remote IMS, IMS recovers the transaction if it is queued for processing or is being processed in the remote IMS.

Recoverable transactions are recovered; irrecoverable transactions are not. After the transaction reaches a commit point, IMS recovers the output response message regardless of the recovery attributes of

the transaction. The discardable and nondiscardable characteristics of the APPC conversation in the originating IMS have no bearing on the transaction's recoverability in the remote IMS across a remote IMS failure.

IMS recovers transactions that are en route to the remote IMS (meaning the transaction message is still en route in the local or intermediate IMS) when the remote IMS fails, regardless of the transaction's recoverability characteristics. After the failure, the remote IMS receives and processes the transactions that were en route at the time of the failure.

Recovering transactions after an intermediate IMS failure

IMS always recovers all messages en route to or from an intermediate IMS that are queued in the intermediate IMS regardless of the recoverability characteristics of the transaction or message.

If the intermediate IMS restarts with either a `COLDSTART`, `COLDCOMM`, or `COLDSYS`, the messages are lost.

Recovering transactions after an application program failure

Transactions sent to IMS from LU 6.2 application programs are processed in the same way as non-LU 6.2 initiated transactions during application program failures.

If an application program fails before reaching a commit point while processing a local or remote transaction from an LU 6.2 device, IMS backs out all messages except those that were inserted to an alternate express PCB and committed with a PURG call. If the failure occurs after the transaction reaches a commit point, IMS recovers everything. If the failing application program's input message was received from another application program (program-to-program switch), this prior application program's processing is still committed (as is true for non-LU 6.2 application programs).

Recoverability flows of LU 6.2 transactions

This topic contains four lists that show synchronous and asynchronous transaction flows, and shows when the transactions are recoverable, irrecoverable, discardable, or nondiscardable.

The following list shows the flow of a transaction sent from an LU 6.2 synchronous conversation to a local IMS.

1. LU 6.2 program: ALLOC LU=*IMS LU name*
2. LU 6.2 program: SEND to local IMS
3. LU 6.2 program: RECEIVE_AND_WAIT
4. Local IMS receives the transaction
5. Transaction is enqueued
6. Transaction executes (if application fails before reaching commit point, message is discarded)
7. Message inserted to I/O PCB
8. Local IMS sends output (Message Control/Error exit routine receives control if LU 6.2 session fails here)
9. Commit point
10. LU 6.2 program: DEALLOCATE

The following list shows the flow of a transaction sent from an LU 6.2 asynchronous conversation to a local IMS:

1. LU 6.2 program: ALLOC LU=*IMS LU name*
2. LU 6.2 program: SEND to local IMS
3. LU 6.2 program: DEALLOCATE
4. Local IMS receives the transaction
5. Transaction is enqueued
6. Transaction executes (if application fails before reaching commit point, message is discarded)

7. Message inserted to I/O PCB
8. Commit point
9. Local IMS: ALLOCATE with TPN=DFSASYNC
10. Local IMS sends output
11. Local IMS: DEALLOCATE

The following list shows the flow of a transaction sent from an LU 6.2 synchronous conversation to a remote IMS:

1. LU 6.2 program: ALLOC LU=*IMS LU name*
2. LU 6.2 program: SEND to local IMS
3. LU 6.2 program: RECEIVE_AND_WAIT
4. Local IMS receives the transaction
5. Transaction is enqueued on remote queue and MSC link (message is recoverable across MSC link failure after enqueue on MSC link)
6. Local IMS sends message across MSC link to remote IMS
7. Remote IMS receives the transaction
8. Transaction executes
9. Output message inserted to I/O PCB
10. Remote IMS enqueues output message to MSC link (message is recoverable across MSC link failure after enqueue on MSC link)
11. Commit point
12. Remote IMS sends output message across MSC link to local IMS
13. Local IMS receives output message
14. Local IMS enqueues output message for LU 6.2 program
15. Local IMS sends output message to LU 6.2 program (Message Control/Error exit routine receives control if LU 6.2 session fails here)
16. LU 6.2 program: DEALLOCATE

The following list shows the flow of a transaction sent from an LU 6.2 asynchronous transaction to a remote IMS:

1. LU 6.2 program: ALLOC LU=*IMS LU name*
2. LU 6.2 program: SEND to local IMS
3. LU 6.2 program: DEALLOCATE
4. Local IMS receives the transaction
5. Transaction is enqueued on remote queue and MSC link (message is recoverable across MSC link failure after enqueue on MSC link)
6. Local IMS sends message across MSC link to remote IMS
7. Remote IMS receives the transaction
8. Remote IMS enqueues the transaction
9. Transaction executes
10. Output message inserted to I/O PCB
11. Remote IMS enqueues output message to MSC link (message is recoverable across MSC link failure after enqueue on MSC link)
12. Commit point
13. Remote IMS sends output message across MSC link to local IMS
14. Local IMS receives output message
15. Local IMS enqueues output message for LU 6.2 program

16. Local IMS: ALLOCATE with TPN=DFSASYNC
17. Local IMS sends output message to LU 6.2 program (Message Control/Error exit routine receives control if LU 6.2 session fails here)
18. Local IMS: DEALLOCATE

Planning for XRF and APPC

All LU 6.2 conversation messages that are sent to a failing IMS are broken at the time of an XRF takeover. New conversations must be allocated on the new active IMS after takeover processing completes. XRF support for LU 6.2 application programs is Class 3 service.

The application program uses the USERVAR parameter as the LU name parameter on the ALLOCATE verb. This allows the same LU name to allocate the conversation on the new active IMS. This is the same concept as the LU usage for non-LU 6.2 IMS terminals, but a different USERVAR name is used. At takeover time, the alternate IMS issues a VTAM MODIFY USERVAR command to redirect the USERVAR from the failed active IMS to the alternate IMS. Also, a VTAM VARY TERM command is issued against the failed active IMS to terminate any conversations held during takeover. Then VTAM propagates the USERVAR changes to other VTAM instances as needed.

Any APPC transactions that are enqueued to a remote IMS are handled by the MSC protocols. After takeover in the local IMS, the queued irrecoverable messages are lost. If the same message is queued to a remote IMS, it is recovered when the local IMS takes over. Processing continues and the reply is sent to the active IMS.

VTAM, z/OS, Tivoli® NetView® for z/OS, and any other product requirements should be carefully reviewed. APPC/IMS requires correct operation of these other products.

Transaction retry characteristics

IMS retries certain abend conditions.

Some examples of these conditions that are retried are:

Deadlock
Lock reject
Fast Path retry conditions

These retry conditions still apply to standard DL/I application programs even if they receive their messages from an LU 6.2 application program. If an abend that can be retried occurs, IMS issues an APPC ATBEXAI call to APPC/MVS to determine if any established conversations exist. If a conversation has been allocated, the abend is not retried and the application program is terminated.

If any CPI Communications resource is being used by the application program, the abend condition cannot be retried. Thus, CPI Communications driven application programs and modified IMS application programs that have allocated an LU 6.2 conversation before the abend occurred can never be retried. This prohibition on retry is necessary, because IMS cannot control the state of CPI Communications resources. IMS supports the DL/I INIT STATUS GROUPB call for CPI Communications driven application programs, but not for modified IMS application programs that have allocated an LU 6.2 conversation before the deadlock is detected.

Qualifying network LU names

You can use the same name for LUs on different systems by adding the network identifier. This eliminates the need to have unique names for every LU on every system in your complex. You can use the network-qualified LU name as the name of the partner LU to allocate remote LU 6.2 conversations and sessions.

A network-qualified LU name consists of a one- to eight-character network identifier of the originating system, followed by a period and the LU name. A network-qualified LU name must be enclosed in single quotes for commands.

Example: `/DISPLAY LUNAME'NETID1.LUAPPC2'`

Use a network-qualified LU name when transmitting data to a remote destination. If no network identifier is present, IMS allows z/OS to determine the destination.

The parameter ALL for either the network identifier or the LU name cannot be substituted in a network-qualified LU name in commands.

Example: `/DISPLAY LUNAME'NETID1.ALL'`

The LU name in the LU 6.2 descriptor can be network qualified. The network-qualified LU name is optional on commands that support the LUNAME keyword.

APPC/MVS uses the name of LU 6.2 network-qualified LUs to allocate conversations. APPC/MVS strips off the network ID and passes the 8-byte LU name to VTAM. Without APPC/MVS support for network-qualified names, the LU name must be unique for the different networks.

The ALL parameter for either the network identifier or the LU name cannot be substituted in a network-qualified LU name in a command.

The LU name in the LU 6.2 descriptor can be network qualified.

Related reading: For information on using network-qualified names in commands, see *IMS Version 15 Commands, Volume 1: IMS Commands A-M*.

Managing multiple LUs for a single IMS system

An IMS system can be represented on a network by more than one LU name. If more than one LU name exists for a single IMS system, you might need to specify which LU should process asynchronous outbound messages.

When more than one LU is defined for an IMS system, one of the LUs serves as the default, or *base*, LU. Normally, the base LU handles all allocation requests for outbound conversations. However, in some cases the remote partner LU might expect the outbound conversation to come from an LU other than the base LU, you can control which LU handles outbound conversation allocation requests. There are three methods for controlling which LU handles outbound conversation allocation requests:

- By specifying an LU name in the OUTBND parameter of the LU 6.2 descriptor in the DFS62DTx member of the IMS.PROCLIB data set. IMS routes outbound messages sent to the LU 6.2 descriptor by using the local LU name specified in the OUTBND parameter.

  Use the `/CHANGE DESC` command to change the local LU for a LU 6.2 descriptor.

  Use the `/DISPLAY DESC` command to display the local LU and other descriptor specifications.
- By specifying an LU name in the OUTBND parameter of the DFSDCxx startup procedure. When an LU name is specified in the OUTBND parameter of the DFSDCxx startup procedure, the specified LU serves as the default LU for all asynchronous outbound conversation messages, regardless of which LU received the original inbound conversation.
- By specifying the local LU option APPCLLU=Y in the DFSDCxx startup procedure. When the local LU option is specified, IMS routes asynchronous outbound conversations through the LU that received the original inbound conversation.
- By overriding any LU name specifications in the LU 6.2 Edit exit routine (DFSLUEE0).

Reassigning an LU to another IMS system

You can reassign an LU from one IMS system to another by using MVS commands.

To reassign an LU to another IMS system:

1. Delete the LU from its current IMS system by issuing the MVS command `SETAPPC LUDEL`.
2. Redefine the LU on the new IMS system by issuing the MVS command `SETAPPC LUADD,ACBNAME=`*`luname`*`,SCHED=`*`new_IMS`*`,NQN`.

DFSAPPC system service

DFSAPPC is an IMS system service for exchanging messages between LU 6.2 application programs (LU 6.2 to LU 6.2), and between LU 6.2 application programs and IMS-managed LTERMs. Message delivery is asynchronous; messages are held on the IMS message queue until they are delivered.

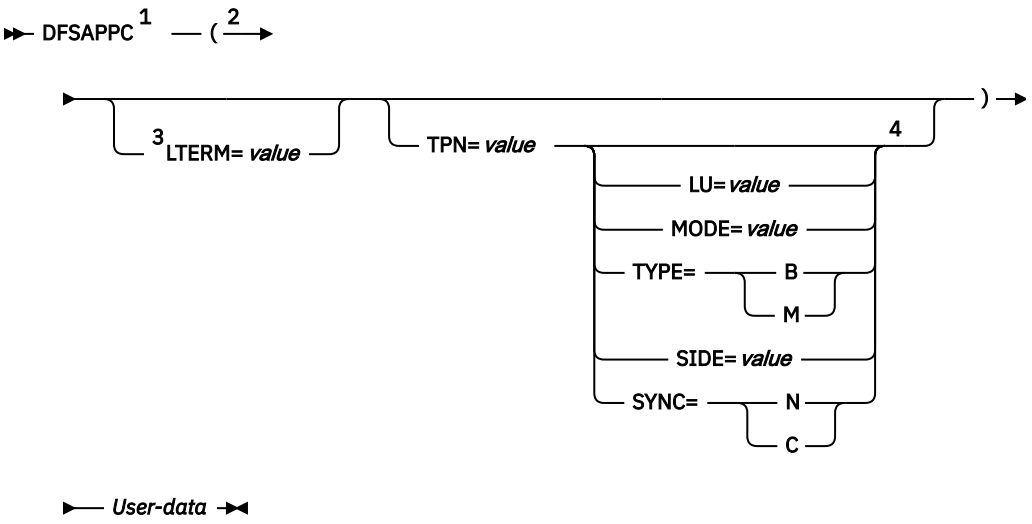
LU 6.2 application program can use DFSAPPC to send messages to IMS-managed LTERMs. Use the LTERM option of the DFSAPPC service to select this capability.

Message switching

Message switching is part of the implicit APPC interface and allows IMS terminals and LU 6.2 application programs to exchange messages. Messages routed to an LU 6.2 application program initiate LU 6.2 application programs.

When using DFSAPPC, the remote device can choose to route a message using either the LTERM or LU 6.2 TPN option. Messages sent with the LTERM option are directed to IMS-managed local or remote LTERMs.[3] Messages sent without the LTERM option are sent to the specified LU 6.2 application program.

The message format for DFSAPPC is shown in the following figure.



Notes:

[1] A mandatory blank is required between DFSAPPC and the options.

[2] Use blanks anywhere within the DFSAPPC options except within keywords or values. Use commas as delimiters between keyword-parameter sets along with or in place of blanks. However, because the TP name character set allows commas, at least one blank must be used to terminate the TPN value.

[3] You can specify either the LTERM= or the TPN= option, but not both. Only use the other keyword options when you specify the TPN= option.

[4] Use the IMS default values for the DFSAPPC options to establish an LU 6.2 conversation with a partner program when the values are not provided by another source. If the DFSAPPC service is coded without specifying any options, use IMS default LU 6.2 conversation characteristics.

Figure 11. DFSAPPC message format

The DFSAPPC option keywords are defined as follows in order of occurrence (except for the keywords following TPN=, which are listed alphabetically):

LTERM=

The 1- to 8-character LTERM option is the name of an IMS LTERM. Messages sent with the LTERM option are directed to an IMS-managed local or remote LTERM. If the LTERM is associated with the

[3] If the LTERM is associated with an LU 6.2 destination, the message is sent as if an LU 6.2 application program had been explicitly selected.

LU 6.2 descriptor, it is treated as if an LU 6.2 application program has been explicitly selected. LTERM names can contain uppercase alphabetic, numeric, and national characters ('@', '$', '#'). When LTERM is specified, other keywords cannot be specified.

TPN=

The 1- to 64-character TPN option is the partner's transaction program name used with the logical unit name to establish an LU 6.2 conversation with a partner program. Because the TP name character set allows commas, at least one blank must be used to terminate the TPN value.

TP names can contain any character from the 00640 character set except a blank. The 00640 character set, documented in the *CPI Communications Specification* includes uppercase and lowercase letters A through Z, numerals 0-9, and 20 special characters.

When the TPN and SIDE options are specified, the TPN name overrides the TP name contained in the side information entry.

Although DFSAPPC allows the use of the 00640 character set, IMS commands do not use this character set. IMS commands can only operate on TPNs that use uppercase alphabetic, numeric, and national characters ('@', '$', '#'). IMS commands cannot operate on extended TPNs.

LU=

The 1- to 17-character LU option is the partner's logical unit name used with the transaction program name (TPN) to establish an LU 6.2 conversation with a partner program.

LU names can contain any character from the APPC/MVS Type A character set. LU names can contain uppercase alphabetic, numeric, and national characters ('@', '$', '#'), and must begin with an alphabetic or national character. You can also use a 17-byte network-qualified LU name in the LU field.

When the LU and SIDE options are specified, the LU name overrides the LU name contained in the side information entry.

MODE=

The 1- to 8-character MODE option is the partner's mode name used with the logical unit name and transaction program name to establish an LU 6.2 conversation with a partner program.

MODE names can contain any character from the APPC/MVS Type A character set. MODE names can contain uppercase alphabetic, numeric, and national characters ('@', '$', '#'), and must begin with an alphabetic or national character.

When the MODE= and SIDE options are specified, the MODE name overrides the mode name contained in the side information entry.

SIDE=

The 1- to 8-character SIDE option is the side information entry name used to establish an LU 6.2 conversation with a partner program.

SIDE names can contain any character from the 01134 character set. The 01134 character set, documented in the *CPI Communications Specification* includes uppercase alphabetic characters A through Z and numeric characters 0-9.

When the SIDE option is specified, the LU, TPN, and MODE options can also be specified to override the values in the side information entry.

SYNC=

The SYNC option allows the application to override the LU 6.2 conversation sync level default provided by IMS.

SYNC=N
Sync_Level is NONE.

SYNC=C
Sync_Level is CONFIRM.

TYPE=

The TYPE option allows the application to override the LU 6.2 conversation type default provided by IMS.

TYPE=B
Conversation type is BASIC.

TYPE=M
Conversation type is MAPPED.

If an error is found while processing the options list, error message DFS1957 DFSAPPC ERROR is sent to the terminal.

Related reading: For more information on Type A character sets, see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*.

Asynchronous output delivery

When creating a message destined for an LU 6.2 application program, IMS establishes conversation characteristics.

These characteristics are extracted from the:

LU 6.2 descriptor
DL/I Change call option list
DFSAPPC message switch options

If IMS cannot extract a particular conversation characteristic from this list, IMS uses the defaults that are shown in the following table. If a side table name is extracted, the default mode name is not used. IMS assumes that side table entries contain a mode table entry name. If an `/ALLOCATE` command for a particular `LUNAME - TPNAME` destination specifies a mode table entry name, that entry name overrides the mode table name specified for the message.

IMS uses the information in the following table to initiate a conversation with the LU 6.2 application program that is associated with the alternate PCB. Certain fields, such as LU name, are application specific. Default values are provided but can be overridden by parameters associated with the message. A default value is used by IMS only if no value is provided by any other source. The application program can modify the default conversation characteristics using an expanded interface to the DL/I CHNG call.

Table 1. APPC/IMS default conversation characteristics

Characteristics	Default value
Conversation_Type	Mapped
Deallocate_Type	Deallocate_Sync_Level
Error_Direction	Receive_Error
Fill	Fill_LL
Log_Data	Null
Log_Data_Length	0
Mode_Name	'DFSMODE' IMS uses the same mode name provided by the inputting LU 6.2 partner to allocate the outbound conversation. That is, whatever mode name the inputting conversation uses, IMS also uses it for outbound allocates.
Mode_Name_Length	7
Partner_LU_Name	'DFSLU'
Partner_LU_Name_Length	5
Prepare_to_receive_type	Prep_To_Receive_Sync_Level
Receive_type	Receive_and_Wait

Table 1. APPC/IMS default conversation characteristics (continued)

Characteristics	Default value
Return_Control	When_Session_Allocated
Send_Type	Buffer_Data
Sync_Level	Confirm
TP_Name	'DFSASYNC'
TP_Name_Length	8

APPC transaction security

The security options for APPC/IMS and LU 6.2 application programs are quite extensive. The partner systems can range from a single-user terminal or workstation to a multi-user system. All systems can have their own complex security environment. Security for IMS can be simple or complex.

Every transaction program name (TPN) must pass a security check before it is executed. The user ID that initiates the transaction is identified on the LU 6.2 format header (FMH5). If no user ID exists because you specify SECURITY=NONE, you can only access resources that are not defined with UACC (NONE). Any TPNs that are accessible in all circumstances should not be defined with UACC (NONE). The TPN security definition is required.

z/OS security consists of two parts. First, z/OS authenticates the transaction user. The LU 6.2 transaction contains security information. The FMH5 contains the user ID, a "profile" name, which is used as the group name, and security options. You supply both the user ID and password. The user ID is defined to RACF, and the password must be valid for the user ID.

If Already_Verified is specified in the FMH5, the sending system verifies the user ID. This user ID must be defined to RACF on the receiving z/OS system. No password is needed in this case.

If SECURITY=NONE is specified, z/OS does no checking. Instead, z/OS builds a special security profile that corresponds to SECURITY=NONE. This allows access to z/OS and APPC/IMS resources that have UACC specified at any level other than NONE. Resources with UACC (NONE) or without a UACC specified cannot be accessed.

After the user ID is established, z/OS verifies that the user ID can execute the specific transaction. z/OS verifies that the user ID's access profile has ACCESS (EXECUTE) for the entity *dbtoken.x.tpname* in the CLASS (APPCTP). The value of *dbtoken* is the dbtoken value specified in the TP_Profile data set. Based on the APPCTPxx parameter specified for this LU, the value of *x* is either the *user ID*, *group* or SYS1.

If either of these security checks fails, z/OS rejects the transaction, and IMS is not informed of it. z/OS can also check:

- Session-level security (RACF resource class APPCLU)
- Port of entry (RACF resource class APPCPORT)
- Local application (RACF resource class APPL)

The IMS administrator should verify that these security checks are successful.

Related reading: For more information on coding the RACF resource classes, see z/OS MVS Planning: APPC/MVS Management.

The security check in IMS is based on the IMS transaction code or executed command name. If the TPN is DFSAPPC, no additional security check occurs. If RACF is used on your system, z/OS rejects the transaction if RACF is not active. The IMS command name or transaction code associated with the TPN is used in the RACF resource class associated with this IMS ('C' or 'T'). RACF checks IMS commands and transactions for all other IMS terminal types in the same way.

If the RACF check is successful, the Transaction Authorization exit routine (DFSCTRN0) is called for transactions, and DFSCCMD0 is called for command authorization. However, the following rules apply to RACF:

- For commands, default security only applies if RACF is not used.
- For remote transactions, RACF is optional.

Otherwise, the exit routines make the security decision.

The intended environment executes APPC/IMS with RACF security active. It is possible to run with RACF not active in the APPC/IMS system, but it is not possible to run with RACF not active in the z/OS system. In this sense, RACF is mandatory for LU 6.2.

The complexity of the security environment is derived partly from the many resources involved (VTAM, z/OS, and IMS) and the granularity of protection that is possible. The security definitions must be closely coordinated for successful operation of the application system.

Related concepts

IMS security (System Administration)

Part 3. Extended Terminal Option (ETO)

These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

© Copyright IBM Corp. 1974, 2020

Chapter 4. Overview of the Extended Terminal Option

The Extended Terminal Option (ETO) of IMS allows you to add VTAM and ISC TCP/IP terminals and users to your IMS without predefining them during system definition.

ETO is part of the IMS Transaction Manager (TM), and provides additional features for users, such as output security, automatic logoff, and automatic signoff.

This topic provides system programmers with the conceptual information that is required to implement and administer ETO. Read the information in this topic if you are unfamiliar with ETO.

Note: ETO is required to define dynamic TCPIP terminals for ISC. However, ETO is primarily used with VTAM terminals. Although some information in the following topics applies to dynamically defined ISC TCP/IP terminals, most of the topics about ETO describe ETO concepts and administration as they relate to VTAM only.

Benefits of using ETO

ETO adds essentially two major enhancements to the Transaction Manager environment. With ETO:

- Users can obtain IMS sessions with VTAM or ISC TCP/IP terminals that have not been defined to IMS during system definition.
- Output messages that are destined for particular users are secure, and they reach only those users.

In addition, by installing ETO, you can achieve each of the following:

- Improved system availability by reducing scheduled down time associated with adding or deleting VTAM and ISC TCP/IP terminals.
- Faster system availability to users, because they can establish an IMS session from any VTAM and ISC TCP/IP terminal in the network.
- Improved IMS security by relating output messages to users, rather than to terminals.
- Reduced number of macros required to define the terminal network. This reduces system definition time and storage requirements.
- Reduced checkpoint and restart time. For ETO terminals and user structures, resources are not allocated until they are actually required; similarly, when they are no longer required, they are deleted.
- Reduced number of skilled system programmer resources that are required for maintaining static terminal definitions.

Related tasks

"Administering the Extended Terminal Option" on page 69
The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

ETO terminology

Certain terms have meanings that are specific to ETO and that are therefore important for understanding and administering ETO.

Terminals

The definitions for terminal, static terminal, and dynamic terminal are described in this topic.

Definitions:

- A *terminal* is a physical VTAM logical unit (LU) that establishes a session with IMS. A physical terminal is represented using a control block.

© Copyright IBM Corp. 1974, 2020

- When terminals are not built by ETO but are defined at system definition, they are called *static terminals*. When messages are sent to a static terminal they are queued to a logical terminal (LTERM) message queue, where they await retrieval by the recipient.
- When a terminal is not defined at system definition and ETO builds a terminal, that terminal is called a *dynamic terminal*, or an *ETO terminal*. For dynamic terminals, the logical terminal (LTERM) is known as a *dynamic user message queue*, LTERM associates the messages with the user, rather than with the physical terminal. Associating messages with the users provides more security for these users, because they can access their messages only when they sign on using their unique user ID. In addition, all users in the network can access their messages from any physical terminal, instead of being restricted to using a particular physical terminal.

Dynamic users

Definition: An ETO *dynamic user* is a user who signs on to a dynamic terminal and who has a unique identification (user ID) that IMS uses for delivering messages. The user is usually associated with a person but can also be associated with another entity, such as a printer.

Terminal structures

A *terminal structure* is an IMS control block that represents a specific terminal that is known to IMS. A terminal structure is created when the individual terminal logs on to IMS. It is deleted when the terminal logs off with no remaining associated activity (such as status that must be retained for the next connection to IMS).

User structures

A *user structure* is a set of IMS control blocks, including a user block and one or more LTERM blocks. The message queues are associated with the dynamic user, as opposed to the physical terminal, and they are queued to the user ID.

The dynamic user structure connects to the physical terminal only when the user signs on. This provides a secure environment, because different users accessing the same terminal cannot receive each other's messages.

IMS creates a user structure when either of the following events take place:

- A dynamic user signs on to IMS.
- Output messages that are destined for a dynamic user are sent to the user, but the user has not signed on to IMS.

Usually, a user structure represents a person who uses IMS. The user structure name is usually the same as the user ID. A user structure can also represent a logical destination, such as a printer. In this case, the user structure name can be the same as or different from the LTERM name that your installation uses in its application programs and its exit routines. For example, you can assign the same name to a user structure for a printer that you assign to its LTERM destination node name. However, output is then queued according to the terminal, and not to the user.

The following figures show the differences between static resources and ETO dynamic resources.

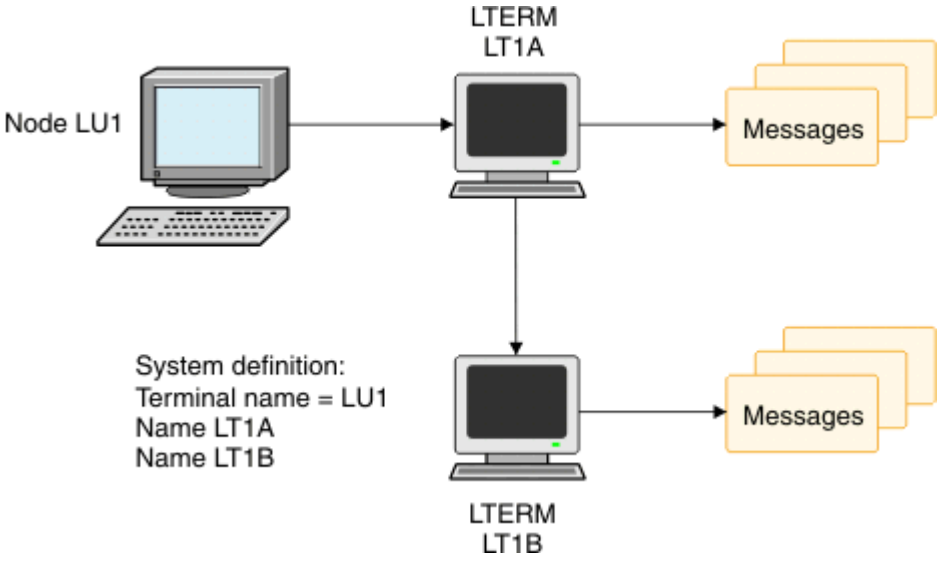


Figure 12. Static resources

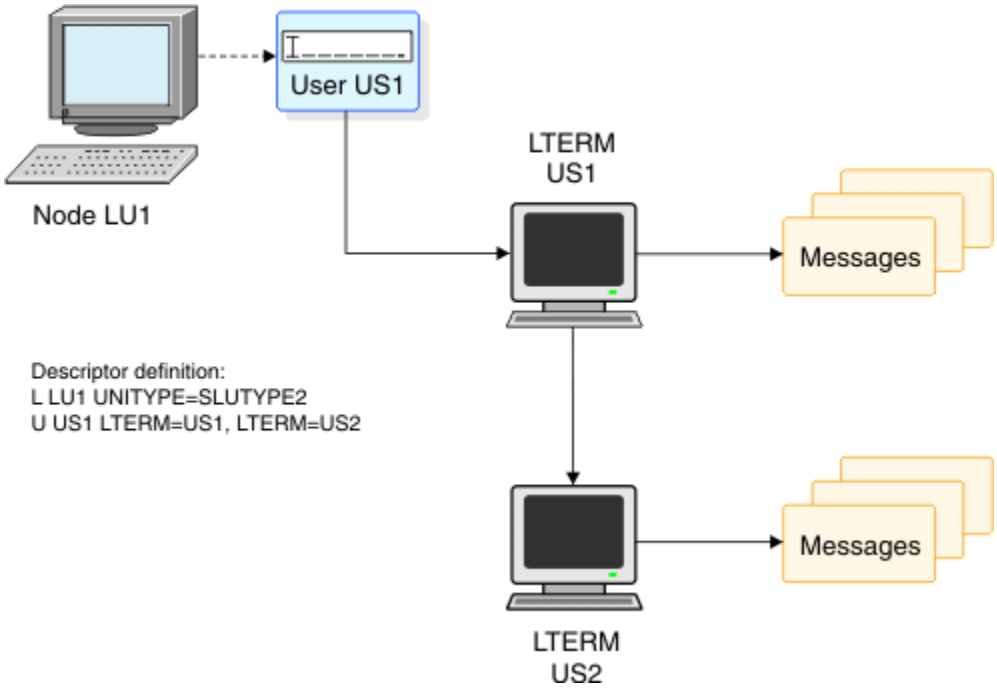


Figure 13. ETO dynamic resources

ETO descriptors

A *descriptor* provides information to IMS when IMS builds a dynamic resource for a logon or a signon. The four types of ETO descriptors are: logon descriptors, user descriptors, MSC descriptors, and MFS device descriptors.

IMS stores descriptors in the following IMS.PROCLIB members:

DFSDSCMx
Contains the descriptors that are automatically generated during IMS system definition. The suffix of DFSDSCMx matches the suffix that your installation specifies on the SUFFIX= parameter of the IMSGEN system definition macro.

DFSDSCTy
Contains customized device descriptors that your installation creates. Descriptors in DFSDSCTy override duplicate descriptors in DFSDSCMx, and the last descriptor that is defined is used.

Logon descriptors

A *logon descriptor* is a skeleton that IMS uses to build an ETO dynamic terminal. It provides information about the physical characteristics of a terminal. IMS uses logon descriptors in conjunction with exit routines to create terminal structures.

The three types of logon descriptors are: generic, group, and specific:

Generic logon descriptor
Provides characteristics for all terminals of a particular type. For example, all SCS printers might share a single generic descriptor. Similarly, all 3270 terminals might share a generic descriptor.

Group logon descriptor
Provides characteristics for a collection of terminals, each of which has compatible hardware characteristics and is defined to IMS in the same manner. The actual characteristics for these terminals are usually identical, but they can differ. IMS uses the group descriptor to derive their characteristics.

Example: You might create separate logon descriptors for different groups of terminals that differ only in the setting for the autologoff (ALOT) time value.

Specific logon descriptor
Provides characteristics for a single terminal, and these characteristics apply only to that terminal. In this case, the descriptor name matches the name of the terminal that it describes.

Note: Although you might need to use specific logon descriptors during the actual migration to ETO, use generic or group logon descriptors after you have migrated to ETO; these kinds of descriptors ease network administration.

User descriptors

A *user descriptor* is a skeleton from which a user structure is built. A user descriptor can provide user options and queue names.

MSC descriptors

An *MSC descriptor* is used to create a remote LTERM, which is an LTERM that does not exist on the local IMS. The physical terminal definition (either static or dynamic) for the remote LTERM is in the remote IMS.

Each MSC descriptor for a remote LTERM is loaded during IMS initialization and tells IMS which MSC link to use for output that is destined for that remote LTERM.

MFS device descriptors

MFS device descriptors enable you to add new device characteristics for MFS formatting without requiring an IMS system definition. The MFSDCT utility (DFSUTB00) uses MFS device descriptors to update default formats in the MFS library.

IMS also uses MFS device descriptors to update the MFS device characteristics table. IMS loads this table only during initialization; therefore, updates are not effective until the next IMS initialization.

Related concepts

"Overview of Multiple Systems Coupling" on page 667

Multiple Systems Coupling (MSC) makes it possible for transactions to be entered in one IMS and processed in another IMS.

ETO concepts

The main purpose of ETO is to dynamically define terminals to IMS. This topic describes such things as ETO terminal and user structures, descriptors and exit routines for dynamic terminals.

When structures are created and deleted

Structures are created in the following situations:

- Log on
- Sign on
- Output is queued to your LTERM
- `/ASSIGN` command is used to assign an LTERM to a non-existent user
- `/ASSIGN` command is used to assign a non-existent LTERM to a user
- `/CHANGE USER` *`username`* `AUTOLOGON` command is directed to a non-existent user

In all cases, IMS searches for an existing structure (terminal or user) before creating a new one.

IMS creates and deletes user structures in the following sequence (This sequence applies only to terminal logon and logoff and to user signon and signoff. When asynchronous output is queued to a user, IMS creates the user structure, as needed.):

1. When you establish a session between IMS and an undefined terminal, IMS selects a logon descriptor.
2. Using the information in the logon descriptor, the customization defaults, and VTAM information, IMS builds a VTAM terminal control block that describes the new terminal.
3. When you sign on, if a user structure does not exist, IMS builds one, using information from a user descriptor that it selects, and then connects this user structure to the terminal structure.
4. IMS deletes terminal or user structures when they are no longer needed to maintain sessions. User structures are typically deleted when you sign off, if no special status needs to be maintained and if no messages remain queued. IMS deletes terminal structures when no terminal status exists (such as trace mode), no user is signed on, and the terminal is idle.

   If you are using Resource Manager and a resource structure, IMS normally maintains status in the resource structure instead of the local control blocks. Therefore, IMS deletes the structures.

   Exceptions: The following terminal structures and user structures are not deleted:

   - SLU P and Finance terminal and user structures are normally only deleted during an IMS cold start if SRM=LOCAL. They can also be deleted, however, if the `/CHANGE NODE COLDSESS` command is used, in which case they are deleted at the first checkpoint following the command.
   - ISC terminal and user structures are only deleted following a cold session termination if SRM=LOCAL.

Descriptors and exit routines

Using descriptors and exit routines, you can assign characteristics to ETO dynamic terminals and assign user structures to be associated with those terminals.

A descriptor provides the basic information for the dynamic terminal. An exit routine completes or changes this information. Two methods of using descriptors and exit routines are:

- You can use many descriptors and code little or no processing logic in exit routines.
- You can use few descriptors and code exit routines to perform much of the processing.

How descriptors are created and used

All descriptors are created during IMS initialization, prior to IMS startup. You must specify that you want the ETO feature support and ensure that the ETO initialization exit routine (DFSINTX0) does not disable ETO.

During IMS initialization, IMS reads and validates all ETO descriptors. IMS initialization then continues, and the descriptors remain in storage for the duration of IMS execution. Any changes you make to descriptors become effective after the next initialization of IMS.

IMS uses descriptors to create both terminal and user structures. IMS rebuilds structures during an IMS restart, if appropriate. For example, if messages are queued for a structure and IMS goes down, the structures are rebuilt when IMS restarts. IMS rebuilds these structures to be the same as they were before the IMS restart. IMS does not use the descriptors or exit routines to rebuild these structures. Therefore, any changes you make to descriptors are only reflected in new structures that are built after IMS restart, and the changes are not reflected in structures that are rebuilt during IMS restart.

Example: USERA signs on using descriptor DESCA which specifies ASOT=20. USERA starts an IMS conversation, and then IMS abnormally terminates. The system programmer changes DESCA to ASOT=10. After the IMS restart, USERB signs on using DESCA. USERA was rebuilt during the IMS restart. USERA still has ASOT=20, and USERB has ASOT=10.

Summary of ETO implementation

The following figure illustrates the ETO concepts and shows an overall view of an ETO implementation.

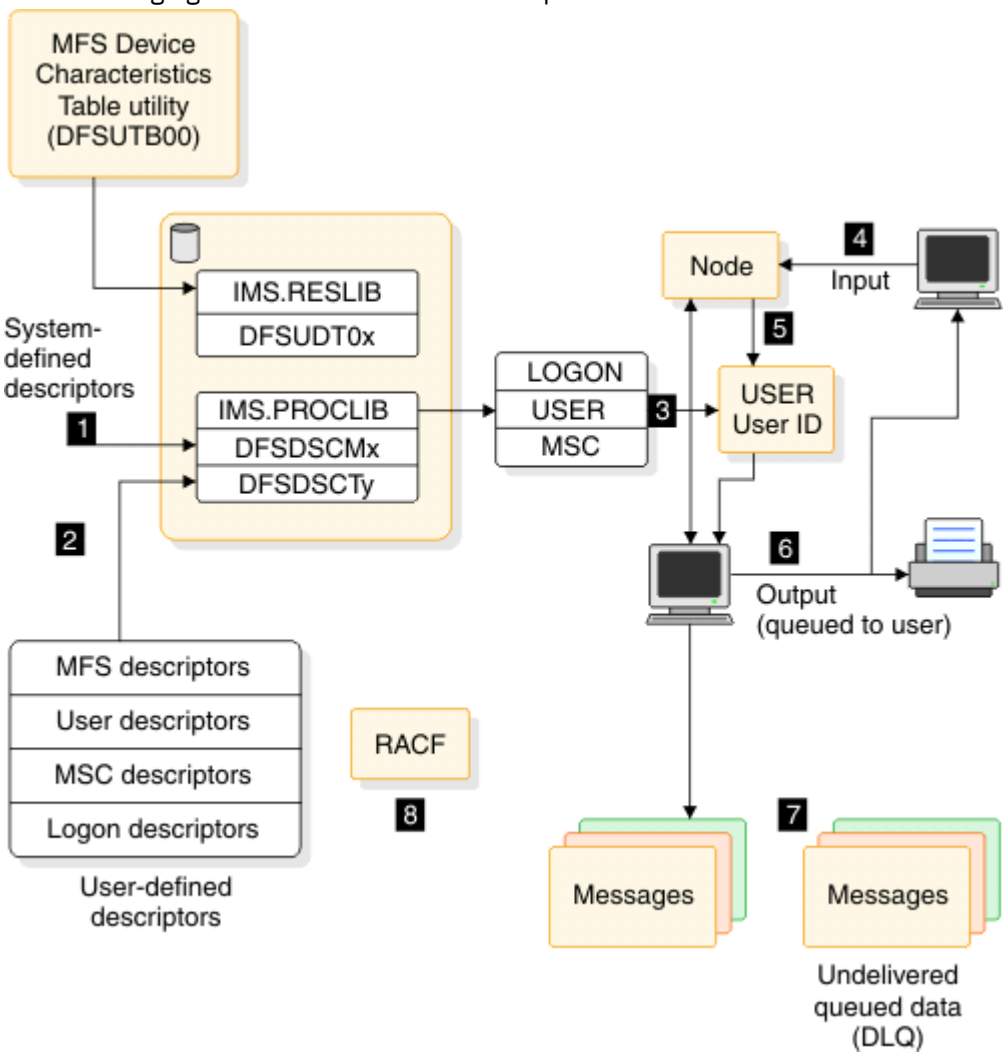


Figure 14. Summary of ETO implementation

- 1 The system-defined descriptors that are built during system definition are stored in IMS.PROCLIB as member DFSDSCMx.
- 2 Your user-defined descriptors that are written to override the system definition defaults are stored in IMS.PROCLIB as member DFSDSCTy. MFS descriptors that are processed by the MFS Device Characteristics Table utility (DFSUTB00) are stored in the device characteristics table.
- 3 Logon, user, and MSC descriptors are loaded at IMS initialization using the input from IMS.PROCLIB.
- 4 The Logon and Logoff exit routines are called during logon and logoff.
- 5 The Signon and Signoff exit routines are called during signon and signoff.
- 6 Output is delivered to the destination specified in the Destination Creation exit routine, unless the user is created during signon.
- 7 If IMS is unable to determine where output should be delivered, the messages are added to the dead-letter queue. Messages might not be delivered because:
  - The user name is not a valid user ID.
  - The user signon is rejected.
  - A physical-to-logical relationship does not exist between the device and the LTERM.
- 8 RACF (or an equivalent product) manages security in the ETO environment.

Chapter 5. Administering the Extended Terminal Option

The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

ETO dynamically creates user structures for a terminal session when:

- The user signs on to IMS.
- Output messages are sent to the user and await retrieval by the user.
- The **/ASSIGN** command is used to assign an LTERM to a non-existent user.
- The **/ASSIGN** command is used to assign a non-existent LTERM to a user.
- The **/CHANGE USER** *username* AUTOLOGON command is directed to a non-existent user.

Some of the administrative advantages of using ETO include:

- You do not need to code the following macros for the system definition stage-1 input stream:

  MSC NAME macros
  VTAM macros: TYPE, TERMINAL, NAME, VTAMPOOL, SUBPOOL

  Removing these macros reduces the complexity of network management.
- You need to perform fewer system definitions.
- You schedule fewer planned outages for new system definitions.

Using ETO, you can ensure that all terminals and users are able to establish sessions with IMS, even if these terminals and users are not defined to IMS during system definition.

You can use execution-time parameters and exit routines to authorize users to access some or all of the functions that ETO provides.

Planning for ETO

Migrating a static-terminal environment to an ETO environment requires planning.

Although you can continue to define VTAM terminals and LTERMs to IMS during system definition, if you do so:

- You cannot take advantage of the ETO features that exist for those terminals.
- You must fully define the terminal. You must supply all TERMINAL macros, NAME macros, and parameters, or use type-2 CREATE commands to supply the information.

ETO terminals must be VTAM terminals.

Restrictions: The following VTAM terminals cannot be ETO terminals:

- IMS master terminal (MTO)
- IMS secondary master terminal
- MSC physical and logical links
- ISC sessions that are used by XRF for surveillance
- LU 6.2 terminals (dynamically created and managed by APPC/IMS)

© Copyright IBM Corp. 1974, 2020

Identifying your requirements

The degree to which you implement ETO across your IMS installation depends on your installation requirements.

ETO is considered fully implemented when no static VTAM terminals exist in the system and when the majority of terminals and users are defined using the default logon descriptor and default user descriptor. However, because installations vary in application program dependencies, the cost of fully implementing ETO also varies.

Your installation should determine the extent to which full ETO feature support is required, based on the following requirements:

Full user-message security
Full implementation of ETO is required for full user-message security. In this environment, no node-name user descriptors exist. Any requirements for user structures that the default user structure does not provide must be defined by user descriptors or by the Signon exit routine (DFSSGNX0).

Dynamic terminal support only
Only partial implementation is required for dynamic terminal support. You can move network definition statements from the system definition to ETO descriptor PROCLIB members. Benefits of this implementation include:

- Fewer system definitions are required in order to maintain network definitions, because you can change descriptors between IMS warm starts.
- Shorter run times are required for system definition, because you do not need to define VTAM terminal networks.
- Improved performance exists for IMS checkpoint and restart, because dynamic terminals and user resources are allocated only when they are used.

With partial implementations, however, you do not achieve improved user-message security, because each LTERM has a fixed relationship with a physical terminal.

ETO restrictions

Before implementing ETO, ETO has a few restrictions that you should be aware of.

The restrictions for ETO include:

- Dynamic terminals are not supported for terminal-related MSDBs or for non-terminal-related MSDBs that have LTERM keys.
- Application programs that use specific LTERM names sometimes require particular ETO customization.

  The DFSUSER user descriptors can help you customize ETO for application programs that have dependencies on LTERM names contained in the I/O PCBs.

Related concepts
“Using DFSUSER user descriptors” on page 86
If IMS does not find a user descriptor that has the same name as the user ID or the terminal that is signing on, and no exit routine has provided one, IMS uses DFSUSER as the default descriptor.

Defining physical terminals

When implementing ETO, to achieve your desired VTAM terminal network you need to be aware of certain requirements and aspects of how ETO and VTAM work together as you plan for and define the physical terminals in the network.

Performing the following actions can ensure that you achieve your desired VTAM terminal network:

- Assess how often IMS application programs depend on specific terminal characteristics.
- Check the accuracy of each VTAM terminal definition. For each dynamic terminal, ETO builds a terminal structure that relies on the VTAM definition for the characteristics (such as the LU type, screen size, and model) for that terminal.

Terminal characteristics that are specified in your IMS system definition might differ from (and override) those in the VTAM definitions. If these terminal characteristics in the IMS system definition are compatible with those of the actual terminal, the discrepancy is not apparent. [4]

- Either provide specific node-name logon descriptors or use the Logon exit routine (DFSLGNX0) for terminals that are not adequately defined using the default logon descriptor.
- Code one of the following two exit routines to determine the device type:
  - The Logon exit routine (DFSLGNX0) can determine the device type by examining the default logon descriptor.
  - The Signon exit routine (DFSSGNX0) can determine the device type later in the process by examining the terminal control blocks.

When receiving a request to establish a terminal session, IMS relies on the following information from VTAM session parameters:

- **UNITYPE**

  The unit type of the node that is attempting to log on. IMS determines the UNITYPE by using the following fields:

  - The LUTYPE field of the CINIT [5] request provides part of the UNITYPE information. The LUTYPE value is usually found in the first byte of the PSERVIC operand of the MODEENT macro, which is used to generate the VTAM mode table entry that is used for a logon. The following figure shows the fields of the PSERVIC operand in the VTAM MODEENT macro.

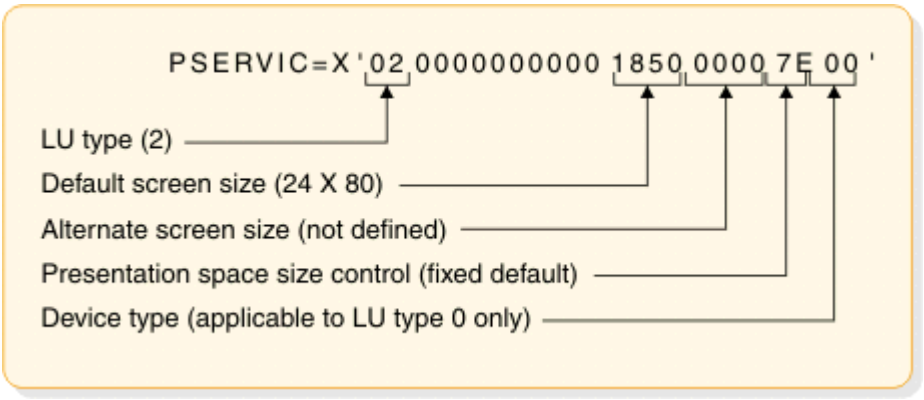


Figure 15. VTAM MODEENT macro PSERVIC operand fields that IMS uses

The following table shows the mapping of the LUTYPE value to the IMS UNITYPE.

Table 2. Mapping for VTAM LUTYPE value to IMS UNITYPE

VTAM LUTYPE	IMS UNITYPE
LUTYPE X'06'	LUTYPE6
LUTYPE X'02'	SLUTYPE2
LUTYPE X'01'	SLUTYPE1 (default) or NTO
LUTYPE X'00'	3270, SLUTYPEP, or 3600/Finance

  - If the LUTYPE field is X'00' (indicating that the terminal is 3270 non-SNA, Finance, or SLU P), IMS must check the transmission services profile specification in the TS field of the CINIT request. The TS value is usually found in the TSPROF operand of the MODEENT macro that is used to generate the

[4] If the actual terminal characteristics do not match those in the IMS definition or the VTAM definition, it is possible that the terminal can function with IMS.

[5] *CINIT* is a network services request sent from a system services control point (SSCP) to a logical unit (LU), asking that LU to establish a session with another LU and to act as the primary end of the session.

VTAM mode table. This LUTYPE value must match the value in the logon descriptor that IMS selects for a logon.

The following table shows the mapping of the TSPROF specification to the IMS UNITYPE.

Table 3. Mapping for TSPROF specification to IMS UNITYPE

TSPROF specification	IMS UNITYPE
X'02' or X'03'	3270
X'04'	SLUTYPEP (default) or 3600/Finance

- **Input RU size**

  The input RU (request unit) size in the BIND must be less than or equal to the RECANY buffer size for the IMS (also required for static terminals).

- **Output RU size**

  The output RU size in the BIND must be greater than or equal to the OUTBUF size that IMS determines for the terminal from the selected logon descriptor. This parameter is also required for static terminals.

- **Screen size and model number**

  For non-SNA 3270 and SLUTYPE2 devices, IMS retrieves both screen size (row and column) and model number from the BIND:

  – For static terminals, the screen size is the value that was specified in the system definition.

  – For dynamic terminals, IMS determines the screen size from the VTAM definition or from the Logon exit routine (DFSLGNX0).

  Recommendation: Until the ETO feature was available, IMS ignored the screen size and model number values in the BIND, because the IMS system definition held this information. Therefore, check to ensure this definition is accurate.

  If you determine that a VTAM definition is inaccurate, you can use the Logon exit routine (DFSLGNX0) to override the VTAM-provided screen size and model number. For example, use a terminal naming convention or MODETAB definition convention. The Logon exit routine can also assign USER=NODE as a name, when appropriate.

  IMS uses the values in the PSERVIC operand of the MODEENT macro, which is used to generate the VTAM mode table entry that is used for a logon. MFS formats must be available for all screen sizes that IMS dynamically builds.

  Restriction: IMS does not use the 3270 Read Partition Query (RPQ) command to determine the screen size from the device controller.

For more information on the Logon exit routine (DFSLGNX0), see *IMS Version 15 Exit Routines*.

Related tasks

“Identifying VTAM device types, screen sizes, and models” on page 74
VTAM logon CINIT user data provides IMS with information to build session control blocks. This information includes logon descriptors, screen size, model numbers, and RU sizes.

Planning for both static and dynamic terminals

Static and dynamic terminals can coexist in the same IMS. However, if users move between static and dynamic terminals, there are situations you need to plan for.

If users move between static and dynamic terminals, plan for the following situations:

- IMS maintains separate queues for static and dynamic terminals. A static terminal has one or more LTERMs associated with it, as controlled by the IMS system definition (or the /ASSIGN command to move an LTERM to a different terminal). Some users become accustomed to having their output queue follow them from ETO terminal to ETO terminal. Static terminals do not provide this feature. Users need to be able to differentiate between static and dynamic terminals, or confusion can result.

- Given one static terminal and one dynamic terminal, separate IMS conversations can exist at the same time. The dynamic terminal conversation belongs with the user structure and follows the user from terminal to terminal. [6] The static conversation belongs to the static terminal and can only be released to a static terminal. This situation is user friendly and predictable only when the user is certain of the terminal type (static or dynamic).

Normally, the terminal operator is able to determine whether a terminal is static or dynamic by checking the security information that is provided at the end of the DFS3650 (SESSION STATUS) message:

OUTPUT SECURITY AVAILABLE
Indicates that the terminal is dynamic, and output is associated with the signon ID.

NO OUTPUT SECURITY AVAILABLE
Indicates that the terminal is either statically defined or that ETO created it by using one of two methods:

- Using a node user descriptor
- Using the Signon exit routine to assign the node name to the user structure

In either case, the output is associated with the terminal, rather than with the user.

Exception: Message DFS3650 might be suppressed if you use the NOTERM option.

Note: You can secure transaction outputs for statically defined VTAM terminals by specifying the **STATICOUTSEC** parameter in the DFSDC*xxx* member of the IMS PROCLIB data set. In this case, outputs are associated with sign-on IDs. If the current user ID does not match the sign-on user ID, the output is discarded.

Recommendation: To ease migration and limit possible confusion, convert to dynamic ETO terminals by using logical groupings within your organization, such as departments or floors.

Defining terminals for growth

When designing your ETO implementation, be sure to plan for growth in your network.

Recommendations:

- To increase future growth potential and system availability, minimize the use of descriptors. The Logon, Signon, and Destination Creation exit routines should provide enough customization, thereby eliminating the need for unique descriptors beyond those that are required for specific terminal types.
- To ensure user data is correctly handled, design exit routines carefully. In particular, carefully plan for user data that is specified during the logon process. Exit routines should work correctly, regardless of whether user data is specified.

  Some terminal types, such as Finance and SLU P terminals, require user data that specifies signon information. If this data is missing, the equivalent information must be provided in the Logon exit routine (DFSLGNX0).

Related tasks

“Identifying VTAM device types, screen sizes, and models” on page 74
VTAM logon CINIT user data provides IMS with information to build session control blocks. This information includes logon descriptors, screen size, model numbers, and RU sizes.

Related reference

Transaction Manager exit routines (Exit Routines)

[6] This assumes that the Signon exit routine (DFSSGNX0) sets the same user name each time, as is usually the case.

Identifying VTAM device types, screen sizes, and models

VTAM logon CINIT user data provides IMS with information to build session control blocks. This information includes logon descriptors, screen size, model numbers, and RU sizes.

The information must be accurate to ensure that the correct terminal-related control blocks is built from information in the logon descriptors. Carefully define your VTAM PSERVIC parameters to ensure that IMS selects the appropriate logon descriptors and establishes screen sizes using specific terminal characteristics.

The BIND is rejected if the input and output RU sizes in the BIND are incompatible with the IMS RECANY and descriptor OUTBUF sizes.

Defining device types

If the definitions are coded incorrectly, IMS chooses the wrong MFS format and device characteristics, possibly causing screen format errors.

IMS dynamic terminal control blocks are built from definitions in a combination of the following:

- The logon descriptor
- Information that VTAM passes to IMS during logon
- The MFS device characteristics table

VTAM passes the physical terminal characteristics to IMS. The following fields have information that IMS uses to determine device types:

- LUTYPE field from the VTAM PSERVIC parameter of the VTAM MODEENT macro
- TS profile from the TSPROF parameter of the VTAM MODEENT macro

Non-SNA 3270 printers and displays

The default descriptor names for non-SNA 3270 printers and displays are DFS327P and DFS3270.

Rules required for defining non-SNA 3270 devices are:

- The TSPROF parameter of the VTAM MODEENT macro must be 2 or 3.
- The PSERVIC parameter of the VTAM MODEENT macro must specify LU Type=0.
- Byte 12 in the VTAM PSERVIC parameter must be modified so that it distinguishes a printer from a video. The following table shows the content that specifies each device type based on the bit location in byte 12:

Table 4. Bits in byte 12 of the VTAM PSERVIC parameter

Bits in byte 12	Content
0-1	Device type: **00** Unspecified device type **01** Printer device **10** Display device **11** Display/printer device (3275)
2-7	Reserved.

If you specify nothing (B'00'), the default is B'10', which is a display device. [7]

[7] B'00' means bits 0 and 1 equal 00. B'10' means bits 0 and 1 equal 10.

Related reading: For more information on defining non-SNA 3270 devices, see *z/OS Communications Server: SNA Programming*.

LU type-2 devices

The following is a list of rules required for defining SLU–2 devices:

- The TSPROF parameter of the VTAM MODEENT macro must be 3.
- The PSERVIC parameter of the VTAM MODEENT macro must specify LU Type=2.

Related reading: For more information on defining LU type–2 devices, see *z/OS Communications Server: SNA Programming*.

3275 devices

VTAM definitions alone cannot identify a 3275 device as it is logging on. Additional information (UNIT=3275) must be specified in a logon descriptor. Define this descriptor in the CINIT user data or in the Logon exit routine. The descriptor itself must be identified as a 3275 device type.

Static 3275 devices are defined in the IMS system definition as follows:

```
TYPE     UNITYPE=3270 TERMINAL UNIT=3275,TYPE=3270-An,SIZE=(24x80),COMPT=PRT1
```

The 3275 has only one buffer. This forces the display and printer components to be the same model. The VTAM definitions for a dynamic 3275 as statically defined are:

```
PSERVIC=X'000000000000185000007EC0'
```

NTO devices

The terminal must identify itself as an NTO device in one of the following ways:

- LU presentation services profile in the BIND image must specify LU1.
- When the LUTYPE is NTO, IMS uses the data stream compatibility byte to precisely define the specific NTO device type. If the LUTYPE specified is SLUTYPE1, the data stream compatibility byte is ignored. The terminal is assumed to be an actual SLUTYPE1 as defined on the logon descriptor.
- Logon exit routine, if available, must accept the DFSNTO (default) logon descriptor or specify an NTO logon descriptor that is defined by your installation using the node name or LOGOND user data parameter.
- The data stream compatibility byte in the CINIT specifies the device type. WTTY indicates a TTY NTO device. If nothing is specified, the device type is LUNS NTO.

LU2 and non-SNA 3270 screen size and model information

After the Logon exit routine approves the logon, BIND image data (screen-size) and feature information from the logon descriptor are used to search the MFS device characteristics table for the appropriate MFS device information.

If the model is specified in the screen size control byte the MFS device characteristics table is not searched. A DFS3646 error message is issued if no match exists on screen-size and feature. The information from the proper MFS device characteristics table entry is then used for that device. This information in the MFS device characteristics table comes from the IMS system definition or the MFSDCT utility (DFSUTB00).

If the screen-size control byte is X'7F' and both the default and alternate screen-sizes are specified, a search of the MFS device characteristics table using alternate screen size commences. If no match is found, another search begins using the default screen size. If no screen size is found, message DFS3646I is issued to the operator.

The screen size (the product of the lines and columns) must be in the range 80-16384. The lines and columns must each be in the range 1-255.

If the screen size and features of the 3270 device that is logging on maps into two or more MFS device characteristics table entries, the first entry in the table that matches the screen size and features is selected.

If the screen-size control byte (the 11th byte of the PSERVIC on the VTAM MODEENT) is X'00' and both default and alternate screen-sizes are specified, a search of the MFS device characteristics table using the default screen-size occurs. If no match is found, another search begins using the alternate screen-size.

Use the Logon exit routine, DFSLGNX0, to override the screen-size or model for the device during logon.

LU2 screen-size and model information

Screen-size and model information applies to LU type–2 devices.

The following is an algorithm IMS uses to determine the model or screen size for ETO terminals:

1. Screen-size is established based on the model specification in the VTAM PSERVIC. IMS determines the model by checking the screen-size control byte in the VTAM PSERVIC field for an X'01', X'02', or X'03'. The model is established accordingly. Screen-size specifications are ignored when the model is specified. X'01' represents a model 1 with a 12x40 screen-size, and X'02' and X'03' represent a model 2 with a 24x80 screen-size. If the model is specified, the MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor.
2. Screen-size is established based on the screen-size specifications in the VTAM PSERVIC. A value of X'7E' in the screen-size control byte causes the default screen size in the VTAM PSERVIC to be used. A X'7F' causes the alternate screen-size to be used first to search the MFS device characteristics table. If no match is found, the default screen-size is used to search; the first match is the screen-size. If no match is found, message DFS3646I is sent to the operator.
3. If the screen-size control byte is X'00', the default and alternate screen-size specifications in the VTAM PSERVIC are used to search the MFS device characteristics table for MFS device information. If a match is made on the default size, the default is used. If a match is made on the alternate size, the alternate is used. If no match is made, the logon is rejected.
4. If the screen-size control byte is X'00' and no screen-size is specified, the device is defaulted to a model 2 device, and the screen-size is established (24x80).
5. The screen-size is established in the BIND parameter in the default screen-size field. The erase write (EW) command is always used.

If you want to override the model value, use the Logon exit routine (DFSLGNX0). Valid values are X'01' and X'02', corresponding to model 1 and model 2 in the logon descriptor. Screen-size and model specifications in the VTAM PSERVIC are ignored. Model=X'01' represents a 12x40 screen-size. Model=X'02' represents a 24x80 screen-size. The MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor.

If you want to override the screen-size value, use the Logon exit routine, DFSLGNX0. Be aware that if you override the model, the screen-size override is ignored. The features from the logon descriptor and the screen-size are used to search the MFS device characteristics table for the MFS device information.

Non-SNA 3270 screen-size and model information

Model information applies to VTAM 3270 Record-Mode devices (non-SNA 3270s).

To determine the model or screen-size for ETO terminals IMS uses the following algorithm:

1. Screen-size is established based on the model specification in the VTAM PSERVIC field. IMS determines the model by checking the screen-size control byte in the VTAM PSERVIC field for X'01', X'02', or X'03'. The model is established accordingly. Screen-size specifications are ignored when the model is specified. X'01' represents a model 1 with a 12x40 screen-size, and X'02' and X'03' represent a model 2 with a 24x80 screen-size. If the model is specified, the MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor. Device type and screen-size are determined from the model value.

2. Screen-size is established based on the screen-size specifications in the VTAM PSERVIC field. A value of X'7E' in the screen-size control byte causes the default screen size in the VTAM PSERVIC field to be used. A X'7F' causes the alternate screen-size to be used first in searching the MFS device characteristics table. If no match is found, the default screen-size is used to search; the first match is the screen-size. If no match is found, message DFS3646I is sent to the operator. Device type and screen-size are implied by the model value.
3. If the screen-size control byte is X'00', the default and alternate screen-size specifications in the VTAM PSERVIC field are used to search the MFS device characteristics table for MFS device information. If a match is made on the default size, the default is used. If a match is made on the alternate size, the alternate is used. If no match is found, the logon is rejected. Device type and screen-size are implied by the model value.
4. If no model information and screen-size are specified in the VTAM PSERVIC field, IMS uses the CINIT's model byte. Model type is established by the VTAM definition statement FEATUR2. The model information applies only if the screen-size control byte is X'00' and if the screen-size is also X'00'. This applies to printers and displays. X'00' is the default and corresponds to a model 1. X'01' corresponds to a model 2. Device type and screen-size are implied by the model value.
5. The screen-size is used to determine the type of write command used. If the screen-size is equal to 960 or greater than 1920, IMS uses erase write alternate (EWA). If the screen-size is less than or equal to 1920 and not equal to 960, IMS uses erase write (EW). The device type and screen size are determined from the model value.

If you want to override the model value, use the Logon exit routine (DFSLGNX0). Valid values are X'01' and X'02', corresponding to model 1 and model 2 in the logon descriptor. Screen-size and model specifications in the VTAM PSERVIC are ignored. Model=X'01' represents a 12x40 screen-size. Model=X'02' represents a 24x80 screen-size. If the model is specified, the MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor. Device type and screen-size are determined from the model value.

If you want to override the screen-size value, use the Logon exit routine (DFSLGNX0). Be aware that if you override the model, the screen-size override is ignored. The features from the logon descriptor and the screen-size are used to search the MFS device characteristics table for the MFS device information. Device type and screen-size are implied by the model value.

Screen definition examples

The following examples show the PESRVIC parameter in the VTAM mode table, what the equivalent TERMINAL macro parameters would be for a static terminal, and the corresponding MFS DEV statement TYPE parameter, which is used for both static and ETO terminals.

- LU0 (non-SNA 3270 video)
  - VTAM mode table: `PSERVIC=X'000000000000000000000200'`
  - TERMINAL macro: `UNITYPE=3270 UNIT=3284/86 MODEL=2`
  - MFS DEV statement: `TYPE=(3270P,2)`
- Model 2 non-SNA 3270 Printer
  - VTAM mode table: `PSERVIC=X'000000000000000000000240'`
  - TERMINAL macro: `UNITYPE=3270 UNIT=3284/86 MODEL=2`
  - MFS DEV statement: `TYPE=(3270P,2)`
- Non-SNA 3270 Display (model specified)
  - VTAM mode table: `PSERVIC=X'000000000000000000000080'`
  - TERMINAL macro: `UNITYPE=3270 MODEL=1`
  - MFS DEV statement: `TYPE=(3270,1)`

  Model information can come from the FEATUR2 parameter for non-SNA 3270; if this parameter is not specified, this is a model 1 (screen-size 12x40). Assume that FEATUR2=1 (specified or default).
- Non-SNA 3270 Display (model specified)

  - VTAM mode table: PSERVIC=X'000000000000000000000080'
  - TERMINAL macro: UNITYPE=3270 MODEL=2
  - MFS DEV statement: TYPE=(3270,2)

  Model information can come from the FEATUR2 parameter for non-SNA 3270; if this parameter is not specified, this is a model 2 (screen-size 24x80). Assume that FEATUR2=2.
- Non-SNA 3270 Display (screen-size specified)
  - VTAM mode table: PSERVIC=X'000000000000185000007E80'
  - TERMINAL macro: UNITYPE=3270 TYPE=3270-A2,SIZE=(24,80)
  - MFS DEV statement: TYPE=3270-A2, where A2=24x80

  For a SLU–2 device, UNITYPE=SLUTYPE2 would be specified (also note the change to the PSERVIC field).

  The screen-size comes from the default screen-size field (24x80). (For a SLU–2 device, the first byte of the PSERVIC field would be X'02'. The last byte would be X'00').
- Non-SNA 3270 Display (screen-size specified)
  - VTAM mode table: PSERVIC=X'000000000000205000000080'
  - TERMINAL macro: NITYPE=3270 TYPE=3270-A3,SIZE=(32,80)
  - MFS DEV statement: TYPE=3270-A3, where A2=32x80

  The screen-size comes from the default screen-size field (32x80). (For a SLU–2 device, the first byte of the PSERVIC field would be X'02'. The last byte would be X'00').
- Non-SNA 3270 Display (model specified)
  - VTAM mode table: PSERVIC=X'000000000000000000000280'
  - TERMINAL macro: UNITYPE=3270 MODEL=2
  - MFS DEV statement: TYPE=(3270,2)

  This is a model 2 non-SNA 3270 display (24x80).
- SNA 3270 Display (model specified)
  - VTAM mode table: PSERVIC=X'020000000000000000000280'
  - TERMINAL macro: UNITYPE=3270 MODEL=2
  - MFS DEV statement: TYPE=(3270,2)

  This is a model 2 SNA 3270 display (24x80).

Planning a high-security environment with ETO

ETO enhances the security of your IMS system. You can customize the ETO security features for your installation needs.

For example, you can customize exit routines that apply to both static terminals and ETO dynamic terminals.

The ETO security features allow you to control each of the following:

- The physical connection of terminals to IMS.
- User signon to IMS.
- Output to users or nodes.
- Message queuing to users. You can customize message queuing two ways:
  - Automatically allocate an LTERM to a user that you identify at signon
  - Use the Destination Creation exit routine (DFSINSX0)
- Command and transaction security, by using RACF (or an equivalent SAF-compliant security product).

Static versus dynamic terminals

You define static terminals during system definition to associate an LTERM with a particular physical terminal. Any user at a terminal can receive output messages that are queued for that terminal, regardless of whether signon is required for that terminal.

The major benefit of ETO is that dynamic LTERMs (output message queues) are managed separately from the terminals and are assigned to a user name. The user can obtain output only after signing on. This dynamic LTERM-to-user association is maintained until the user signs off, and it remains after signoff if IMS does not delete the user structure.

Planning for MFS

After you implement ETO dynamic terminal support, the number and diversity of terminal types is likely to increase. When a terminal dynamically establishes an IMS session, MFS formats might not be available for the device type that is requesting the session.

You can solve this problem by:

- Restricting device types to only those that are used in the MFS definitions for your application programs.

  Recommendation: Use the Logon exit routine (DFSLGNX0) to select the desired logon descriptor.
- Extending the MFS device output formats to include the new terminal types that connect to the IMS.

  Recommendations:

  – Use the MFS Generation utility (MFSGU) facilities, such as the STACK statement, to make creating these additional formats easier.

  – Use the MFSDCT utility (DFSUTB00) to avoid needing to perform an IMS generation.

Often, multiple device types have similar or identical capabilities, so the distinction of device type might not be important to your environment. Although IMS application programs are usually not sensitive to device type, such a dependency can exist. For example, MFS can create different input messages from the same input data stream, due to differences in the format specification for device input.

Planning user IDs

When planning user IDs for use with ETO there are a number steps you should take.

The steps include:

- Ensure that dynamic user structures have unique names in IMS. Support for unique user IDs depends on whether user-based security is important. For user-based security, user IDs must be unique to the user. Otherwise, user IDs can be the same as the terminal LU name.
- Code the Signon exit routine (DFSSGNX0) to provide user ID suffixing, if users need to sign on to more than one terminal with the same user ID at the same time. Review and select the SGN and RCF EXEC keyword parameters.
- Analyze how application programs use the user field in the I/O PCB.

Planning user queue names

When planning user queue names for use with ETO there are a number steps you should take.

When planning user queue names:

- Ensure that dynamic user queue names are unique in the IMS. Support for unique queue names depends on whether user-based security is important. For user-based security, user queue names must be unique to the user. Otherwise, user queue names can be the same as the terminal LU name.
- Specify user descriptors or provide logic in the Signon exit routine (DFSSGNX0) if the user queue name cannot be the same as the user name.
- Analyze how application programs use the LTERM field in the I/O PCB.

Planning operations

When planning for operations there are a number of steps you should perform.

The steps you should take when planning for operations include:

- Update your MTO procedures to reflect the concept of dynamic resources. The MTO needs to become familiar with the command input fields and response formats.
- Update the help-desk procedures to reflect the IMS terminal and user resource structures, as well as the commands that are required to diagnose user's problems.
- Review and update automated-operator procedures, if necessary.
- Develop procedures for handling the dead-letter queue.
- Identify requirements for autosignoff and autologoff.
- Develop standards for the conditions under which system-wide values should be used and when they should be overridden.

Planning for MSC support with ETO

You can define MSC physical and logical links by using static system definition macros or you can create them dynamically by using type-2 CREATE commands.

You can identify a remote MSC NAME to IMS using an MSC descriptor. The descriptor relates each remote resource to the link path name of a generated MSNAME macro.

Recommendation: Define remote LTERMs with MSC descriptors to maintain consistency between remote and local systems.

If you choose not to use MSC descriptors in order, the MSC Verification utility (DFSUMSV0) recognizes the remote LTERMs but not the corresponding local LTERMs in the target system. In this case, IMS issues warning message DFS2331W.

IMS processes MSC descriptors that are associated with MSC links defined within the IMS. IMS ignores other MSC descriptors. You can maintain a single network definition for remote LTERMs in IMS.PROCLIB for multiple interconnected IMS systems that use MSC. In this case, each logical link path name must be unique throughout the entire network.

Coding ETO descriptors

IMS uses ETO descriptors to dynamically build terminal structures and user structures.

The four types of ETO descriptors are:

Logon
User
MFS device
MSC

The basic format for all ETO descriptors is:

Column
Description

1
One of the following descriptor types:

L
Logon descriptor

U
User descriptor

D
MFS device descriptor

M
MSC descriptor

*****
Comment line (ignored by IMS)

3-10
Name of the descriptor, using the following conventions:

- Must be one to eight alphanumeric characters.
- For logon and user descriptors, characters are limited to A-Z, #, $, and @.
- For MSC descriptors, use the link name.

12-72
One or more keywords and their parameters, separated by blanks. Use commas to separate multiple parameters for a single keyword.

73-80
Optional sequence numbers (ignored by IMS).

Related concepts
ETO descriptors (System Definition)

Creating descriptors using the system definition process

To ease migration, you can create a starter set of ETO descriptors (except MFS descriptors) using the ETOFEAT keyword on the IMSCTRL macro at system definition. You can add to or modify the starter set by creating an additional IMS.PROCLIB member.

For MFS descriptors, the system definition creates a device characteristics table based on the stage-1 input that can be used as one of the inputs to the MFSDCT utility (DFSUTB00). The device characteristics table contains what the system-defined device characteristics table contains, plus any additional entries from device descriptors.

Creating descriptors during the system definition process saves you time and ensures that the descriptors are correct. IMS generates an ETO descriptor report, which provides information on the relationship between ETO descriptors and the IMS system definition resources that they represent.

Related concepts
Including IMS ETO in the IMS system (System Definition)
Related reference
IMSCTRL macro (System Definition)

Storing descriptors

Descriptors that are created during system definition are stored in the IMS.PROCLIB member, DFSDSCMx. Subsequent system definitions of the same stage-1 input deck overwrite the DFSDSCMx member.

Recommendation: To avoid losing descriptors when member DFSDSCMx is replaced, store descriptors that you create by using TSO or z/OS utilities in IMS.PROCLIB member, DFSDSCTy. If you need to update descriptors that are created at system definition in DFSDSCMx, use TSO or a z/OS utility (such as IEBUPDTE) to make the updates.

Creating logon descriptors

Logon descriptors provide IMS with information about the physical characteristics of the terminals that establish logon sessions. These characteristics must be consistent with the VTAM logon BIND characteristics.

This topic describes how to create and use logon descriptors.

Creating logon descriptors during system definition

When you specify IMS system definition options to create ETO logon descriptors, IMS dynamically creates a logon descriptor for every unique VTAM TERMINAL macro that you have specified (For each TERMINAL macro definition, a node user descriptor is also created for use as a migration step to ETO).

IMS system definition can produce up to 37 common logon descriptors for each device type. The descriptor that defines the largest number of terminals of that type becomes the default logon descriptor. This logon descriptor assumes the IMS-defined default name for that type. The other terminals of that type create their own unique logon descriptor by using a suffix on the default name.

For terminal definitions that do not match one of the 37 common descriptors, IMS creates an individual logon descriptor.

These descriptors are generated as comments (with an asterisk in column 1). For example, `*L3270A`. To choose a descriptor that you need, remove the asterisk.

The naming convention for the 37 common logon descriptors that are created during the system definition process is:

- The last character of the name must be unique.
- Blank for the most common, then 0-9 and A-Z.

Restrictions:

- During system definition, IMS does not create ETO logon descriptors for the primary or secondary master terminal, or for LU 6.1 terminals that are defined as XRF ISC links.
- The following keywords are not supported on logon descriptors: PU, SIZE, MODEL, TYPE, MSGDEL.

Criteria for selecting logon descriptors

The logon descriptor contains data that is related to the node and the VTAM CINIT. This data allows IMS to create a control block structure that supports a session.

IMS uses the following criteria (in sequence) to select a logon descriptor:

1. IMS uses existing control blocks for terminals. IMS does not look for a descriptor if it finds existing control blocks.
2. IMS determines whether the Logon exit routine is used to define logon descriptor names.

   For VTAM terminals, the exit routine extracts the name from the VTAM CINIT user data, or another appropriate algorithm. The LUTYPE and TS= (transmission service level) fields in the VTAM CINIT data must agree with the selected descriptor; otherwise, the logon is rejected. IMS rejects invalid logon descriptor names.
3. In response to a request to establish a session (logon), IMS examines the LOGOND parameter. The LOGOND parameter can indicate the logon descriptor for IMS to use.

   LOGOND is a parameter in the VTAM CINIT user data. LOGOND is also a keyword on the IMS **/OPNDST** command.

   Restriction: LOGOND is not valid for ISC parallel sessions that use VTAM.
4. If IMS does not find a logon descriptor name, it looks for a logon descriptor with the same name as the VTAM CINIT LUNAME.

   Recommendation: Use the node name on the logon descriptor if you expect any of the following situations:

   - You do not expect to add more of the same terminal type.
   - You do not have many of the terminal type.
   - You want to simplify the logic in the Logon exit routine.

- For ISC, you want to specify parameters (such as OUTBUF=) other than the default ISC logon descriptor, and you have not coded an exit routine.

If these criteria do not yield a valid logon descriptor name, IMS selects a descriptor by using the default criteria.

Criteria for selecting a default logon descriptor

IMS provides a default logon descriptor for each VTAM terminal type. The logon descriptor name is based on the LU type and TS profile.

If necessary, you can add logon descriptors. If you do not provide a logon descriptor, and IMS does not locate one in IMS.PROCLIB or in the Logon exit routine (DFSLGNX0), IMS uses the algorithm shown in the following table to assign a default logon descriptor name. The VTAM LUTYPE, IMS UNITYPE, and IMS default logon descriptor names are shown for each terminal type.

The default logon descriptor is determined in one of three ways:

- IMS looks for the most common logon descriptor created during IMS system definition and uses it as the default.
- If no terminal definition exists for that terminal type, IMS creates a default logon descriptor.
- Your installation can create a default logon descriptor.

Table 5. Mapping for VTAM LUTYPE, IMS UNITYPE, and default logon descriptor names

VTAM LUTYPE	IMS UNITYPE	Default descriptor name
X'06'	LUTYPE6	DFSLU61
X'02'	SLUTYPE2	DFSSLU2
X'01'	SLUTYPE1 (Default)"1" on page 83 NTO	DFSSLU1 DFSNTO
X'00' (TS = X'04')	SLUTYPEP (Default)"2" on page 83 Finance 3601	DFSSLUP DFSFIN DFSFIN
X'00' (TS = X'02' or X'03')	3270 3270, UNIT=3284	DFS3270 DFS327P

Notes to table:

1. IMS does not distinguish between SLUTYPE1 and NTO terminals; SLUTYPE1 is the default. To support both SLUTYPE1 and NTO terminals, do the following:
   - Override the DFSSLU1 default logon descriptor by using the Logon exit routine (DFSLGNX0) or by specifying the LOGOND parameter.
   - If your installation has no SLUTYPE1 terminals, rename the DFSNTO logon descriptor to DFSSLU1 in order to make it the default for LU type X'01' terminals.
2. IMS does not distinguish between SLU type-P, Finance, and 3601 terminals; SLUTYPEP is the default. ETO considers Finance and 3600 series terminals to be the same. To support both SLUTYPEP and 3600/Finance terminals, do the following:
   - Override the DFSSLUP default logon descriptor by using the Logon exit routine (DFSLGNX0) or by specifying the LOGOND parameter.
   - If your installation has no SLUTYPEP terminals, rename the DFSFIN logon descriptor to DFSSLUP in order to make it the default for LU type X'00' (TS = X'04') terminals.

Using NTO, 3600/Finance terminals

Because of conflicting CINIT information, IMS cannot generate DFSNTO or DFSFIN as a default logon descriptor name. CINIT parameters for SLU 1 are identical to NTO, and SLU P is identical to 3600/Finance.

Recommendations: For 3600/Finance and NTO terminal types, take each of the following actions:

- Always supply the appropriate logon descriptor name in the Logon exit routine (DFSLGNX0). You can generate this name as a constant, based on LU name, LU type, or CINIT user data.
- Ensure that the logon descriptor name is provided as the LOGOND parameter from the BIND user data.
- Ensure existence of a logon descriptor that matches the node name of the terminal that is logging on. If IMS does not find a default logon descriptor, the logon attempt fails.

Recovering ETO terminals using XRF

To recover ETO terminals in an XRF environment, use the BACKUP= parameter when specifying the logon descriptor.

The BACKUP= parameter sets the priority that controls the order in which sessions are switched.

When 3600/Finance and SLU-P terminals are defined as class–2 terminals in an XRF environment, automatic re-logon and re-signon occur during a takeover.

Related reading: For general information on XRF, see *IMS Version 15 System Administration*.

Creating user descriptors

User descriptors provide information relating to user options and user structure names. IMS needs user descriptors in order to create control blocks that enable users to use ETO terminals.

The three types of user descriptors are:

Installation-created
Node user
DFSUSER

IMS chooses the first valid descriptor, using the given sequence. IMS creates DFSUSER and node user descriptors using system definition options.

This topic describes how to create user descriptors.

Creating user descriptors during system definition

User descriptors are generated from each VTAM TERMINAL or VTAMPOOL SUBPOOL macro.

User descriptors that are created during system definition by using the VTAM TERMINAL macro have the same name as the terminal. User descriptors that are created by using the VTAMPOOL SUBPOOL macro have the same name as the subpool.

For user descriptors that are created by using a SUBPOOL macro, you cannot set a response option (TRANSRESP, NORESP, FORCRESP), because it is defined on the TERMINAL macro for static definitions. You need to add the response option (appropriate for your installation) to any user descriptor that is created with a SUBPOOL macro.

Criteria for selecting user descriptors

When a user signs on, if the user structure does not exist, IMS selects a user descriptor to build the user structure.

IMS selects the user descriptor according to the following criteria:

1. An installation-written exit routine can select the name of the user descriptor (user ID, node name, or DFSUSER).

2. IMS looks for USERD, provided at logon. IMS also looks for a descriptor name (user ID, node name, or DFSUSER) that is specified in the USER descriptor field of the **/SIGN** or **/OPNDST** command.
3. IMS looks for a descriptor that has the same name as the user ID (installation-created user descriptor only). If IMS finds one, and an LTERM keyword is not specified in the descriptor, IMS creates a user structure and a single LTERM, both of which have the same name as the user ID.
4. IMS looks for a descriptor that has the same name as the VTAM node (node user descriptor). If IMS finds one, and an LTERM keyword is not specified in the descriptor, IMS creates a user structure and a single LTERM, both of which have the same name as the VTAM node. However, no output security is associated with this user structure. Any user that signs on at a terminal can receive messages that are queued for that terminal.
5. IMS selects the default user descriptor, DFSUSER. IMS creates a user structure and a single LTERM, both of which have the same name as the user ID.

Using installation-created user descriptors

You can create your own installation-specific user descriptors that meet important criteria for your site.

The name of the installation-created user descriptor is the same as the user ID. You can add values to these user descriptors that are not provided with node user descriptors or DFSUSER descriptors.

Using node user descriptors

Node user descriptors help in migrating from static terminal definitions to ETO dynamic terminal definitions.

During IMS system definition, node user descriptors are created as an option. Node user descriptors can be useful when exit routines that perform descriptor selection are not yet complete.

Node user descriptors must have the same name as their associated terminals. Therefore, IMS creates unique node user descriptors that retain user options and user structure names that exist in the system definition. When IMS system definition creates ETO descriptors, a node user descriptor is created for each terminal that has a VTAM TERMINAL macro or VTAMPOOL SUBPOOL macro. Node user descriptors are created even when they match the DFSUSER options (No descriptor is created for an ISC terminal that is defined for parallel session support).

If the LTERM parameter of the node user descriptor contains the same name as a statically defined LTERM, the node user descriptor is ignored. IMS issues message DFS3673W. Consequently, all node user descriptors that are created during system definition are generated as comment statements. To use the node user descriptors, remove the asterisks.

In most cases, node user descriptors are not needed. You only require node user descriptors when the default user descriptor (DFSUSER) or the Signon exit routine (DFSSGNX0) cannot supply the user options you desire. Using a TSO or a z/OS utility, you can discard most of the node user descriptors that are created during system definition.

Be aware that when you use node user descriptors, you cannot use ETO's ability to eliminate the need for predefining terminals that connect to IMS. Using node user descriptors lose output security for each user as well.

Recommendations:

- If continuous availability of IMS is critical, eliminate node user descriptors as soon as possible. Because adding new terminals by using node user descriptors requires an IMS restart, use exit routines instead of node user descriptors.
- To avoid using node user descriptors, use the Signon exit routine (DFSSGNX0) to have IMS build a user structure using the node name as the user name. Although this is similar to using node user descriptors, it avoids predefining large numbers of these descriptors at IMS initialization.
- You can use node user descriptors to start a session when output is available, using **/OPNDST**, LOGON, or autologon. To use autologon, specify the SHARE option on the TERMINAL macro.

- If you select a default user descriptor using the Signon exit routine (DFSSGNX0), you can set a bit to indicate that IMS should create the user structure using the node name for the user structure name. This is the same as selecting a node user descriptor. By creating node-name structures by using the Signon exit routine, rather than by defining a node user descriptor for each terminal, large network installations benefit from decreased complexity.

Using DFSUSER user descriptors

If IMS does not find a user descriptor that has the same name as the user ID or the terminal that is signing on, and no exit routine has provided one, IMS uses DFSUSER as the default descriptor.

IMS uses DFSUSER for both signon and application program output processing. Using DFSUSER also enables IMS to dynamically create a user structure for a signon request when no other user descriptor is available.

IMS creates DFSUSER when you specify the system definition options to create ETO descriptors. DFSUSER contains the options specified most frequently in your IMS system definition, and it should satisfy most users. However, using the DFSUSER descriptor does not require you to use all of these options. You can code the Signon exit routine (DFSSGNX0) to make changes to the structures that are built by using DFSUSER.

DFSUSER builds a user structure that has the same name as the user ID that is specified on the **/SIGN** command. IMS allocates to this user structure a single LTERM, which also has the same name as the user ID. You can use the Signon exit routine (DFSSGNX0) in order to supply multiple queues, if necessary.

Recommendation: Fully implementing ETO involves creating most user resources using DFSUSER. After migrating to ETO, use the DFSUSER descriptor for as many users as possible. Minimizing the number of descriptors reduces the administrative workload of an IMS network.

The user structure name becomes the user ID. When you use the DFSUSER descriptor, you can modify user structure names and other options by using the Signon exit routine (DFSSGNX0).

If an application program has a dependency on an LTERM name that exists in an I/O PCB, you can customize ETO in one of two ways:

- Provide a user descriptor for each user that is to use a particular application program.
- Use the Signon exit routine (DFSSGNX0) to tailor the default action for a subset of the user population.

The Signon exit routine (DFSSGNX0) and the Destination Creation exit routine (DFSINSX0) enable you to dynamically create user structures, even when the specified user descriptors do not contain data to create LTERMs.

Related reference

Signon exit routine (DFSSGNX0) (Exit Routines)

Destination Creation exit routine (DFSINSX0) (Exit Routines)

Creating MFS device descriptors

With MFS device descriptors, you can define screen size and feature combinations that are not generated during IMS system definition.

The MFSDCT utility (DFSUTB00) uses the MFS device descriptors in order to update device type, screen size, and features in the MFS device characteristics table. MFSDCT also uses MFS device descriptors to generate new MFS default formats, without requiring changes to the system definition. The MFS device descriptors are not created as part of a system definition, but are instead an optional step after IMS system definition.

MFS device descriptors define terminal characteristics of dynamic terminals that differ from static terminals. Unless you use exit routines to override screen sizes, you need to ensure that all ETO terminal screen sizes that are different from those that are defined for static terminals are defined using MFS device descriptors.

Recommendation: If you have users who log on to terminals that have different device characteristics, you need to create an expanded set of MFS formats. Input and output messages that are delivered to non-originating terminals are formatted to the terminal according to the MFS formatting specifications that are defined for that device.

Building the device characteristics table

When someone logs onto an ETO 3270 or SLU-2 terminal that is connected to IMS, usually the screen size is provided to IMS in the VTAM CINIT PSERVIC data. Sometimes the model number is also provided.

IMS uses this screen size and the features that are specified on the logon descriptor in order to search the MFS device characteristics table, which contains one or more entries. Each entry identifies the following:

The MFS device type
The screen size
The features

Searching the table, you can find the MFS device type. If the search is unsuccessful, the logon attempt is rejected, and messages DFS3646I and DFS3672I are issued.

The MFS device characteristics table is built in one of two ways:

- The IMS system definition process builds an MFS device characteristics table based on the TYPE=, SIZE=, and FEATURE= specifications of the TERMINAL macro. Each unique combination has an entry in the MFS device characteristics table.
- The MFSDCT utility (DFSUTB00) builds or modifies an MFS device characteristics table based on the input of the MFS device descriptor. These descriptors have the TYPE=, SIZE=, and FEATURE= specifications that the TERMINAL macro has. When the VTAM CINIT PSERVIC indicates a model number instead of a screen size, the MFS device characteristics table is not searched. A model number indicates a certain screen size and MFS device type.

The MFS device characteristics table is not searched if the VTAM CINIT PSERVIC field contains a model number instead of a screen size.

Related reading: For more information on using the MFSDCT utility (DFSUTB00), see *IMS Version 15 System Utilities*.

Using the MFSDCT utility (DFSUTB00)

IMS searches the device characteristics table in order to locate MFS device information during session initiation only in certain situations.

IMS searches the device characteristics table during session initiation in the following circumstances:

- The screen size is overridden.
- The model value in PSERVIC is X'00', X'7E', or X'7F'.

The MFSDCT utility (DFSUTB00) enables you to define screen sizes that are not defined during IMS system definition. The MFSDCT utility uses MFS device descriptors in PROCLIB member DFSDSCMx and DFSDSCTy, without performing an IMS system definition. The new screen size definitions are added to those that are already defined.

The MFS utility does the following:

- Reads device descriptors from IMS.PROCLIB members DFSDSCMx and DFSDSCTy.
- Builds one device descriptor table entry statement for each new device descriptor.
- Terminates if no descriptors are specified.
- Optionally loads the existing device characteristics table from IMS.SDFSRESL, and builds one entry statement for each existing table entry.
- Passes the table entry statements and the DCTBLD and MFSINIT macros as input to assembler.

- Prepares assembler output as a new or updated device characteristics table, as well as a new set of default MFS format definitions. Output is in separate files for subsequent processing.

The MFSDCT utility procedure is found in IMS.PROCLIB. You must evaluate the screen size requirements and code MFS device descriptors to meet those requirements. The MFSDCT utility must generate all possible combinations of screen sizes and features that your installation might require.

To use the MFSDCT utility, follow these steps:

1. Execute DFSUTB00 (the MFSDCT utility).
2. Assemble the new device characteristics table.
3. Link edit the new device characteristics table into IMS.SDFSRESL.
4. Execute phase 1 of the MFS Language utility in order to generate new default MFS format control blocks.
5. Execute phase 2 of the MFS Language utility; the new default MFS formats are loaded into IMS.FORMAT.

Problems with the device characteristics table are often indicated by error messages DFS3646I and DFS3672I during logon processing.

Related reference

MFS Device Characteristics Table utility (DFSUTB00) (System Utilities)

Creating MSC descriptors

MSC descriptors relate remote LTERMs to statically defined MSC links.

IMS creates MSC descriptors when you specify IMS system definition options to create ETO terminal descriptors. MSC descriptors relate remote NAME macros to defined MSC links.

Recommendation: Define all LTERMs in remote IMS systems by using MSC descriptors. The MSC Verification utility in the target IMS system can then associate the remote LTERMs with the corresponding local LTERMs. IMS issues message DFS2331W if the corresponding local LTERMs are not found.

IMS processes only those MSC descriptors that are associated with the MSC links that are defined within the system being initialized. IMS ignores all other MSC descriptors. If each logical link path name is unique in the network, you can maintain your network's MSC definition source in a single PROCLIB member, IMS.PROCLIB.

Exit routines

You can customize ETO with a variety of exit routines delivered with IMS.

You can use the following exit routines to customize ETO:

- Initialization exit routine (DFSINTX0)
- Logon exit routine (DFSLGNX0)
- Logoff exit routine (DFSLGFX0)
- Signon exit routine (DFSSGNX0)
- Signoff exit routine (DFSSGFX0)
- Destination Creation exit routine (DFSINSX0)
- Greetings Message exit routine (DFSGMSG0)

If ETO is enabled, these exit routines are loaded during IMS initialization. All non-MSC and non-LU 6.2 VTAM terminals (static or dynamic) can use these exit routines.

You can code these ETO exit routines with a wide range of processing logic for any of the following purposes:

- Enforcing naming conventions

- Selecting user queues
- Overriding terminal characteristics, such as screen size
- Enhancing security by limiting access to IMS terminals
- Overriding security by allowing signon without using a security product, such as RACF
- Selecting operational parameters, such as timeout values

Related reference
Transaction Manager exit routines (Exit Routines)

Starting ETO

To include ETO in your IMS system, specify ETO=Y in the IMS or DCC startup procedure.

The default for the ETO= keyword in the startup procedures is ETO=N. If ETO=N remains specified, IMS rejects requests to establish sessions for undefined terminals. Messages that are destined for nonexistent queues are refused, and IMS issues message DFS064 or an A1 status code.

When IMS initializes, it calls the Initialization exit routine (DFSINTX0). You can code this exit routine to disable ETO, or to create and load data that you want ETO to use. IMS maintains a pointer to this data and passes this pointer to the ETO exit routines as an input parameter. IMS passes this pointer to non-ETO exit routines through the SCDINTXP field in the system contents directory (SCD).

When you enable ETO, IMS validates each ETO descriptor:

- If a descriptor is coded incorrectly, IMS ignores it and issues message DFS3640W to the MTO.
- If IMS detects an invalid parameter within a valid descriptor, IMS substitutes the default parameter, and processing continues with message DFS3641W.
- If IMS is unable to read all the descriptors (for example, because of read errors), IMS abends with abend U0015. IMS does not abend if it finds one valid logon descriptor and one valid user descriptor.
- IMS does not start a system whose descriptor information is incomplete.

Descriptors, exit routines, and the MFS device characteristics table can be added, deleted, or updated before IMS initialization time. However, if a control block for a session exists across restart, a change in the descriptor that built the control block does not take effect until after the control block is deleted. For example, 3600/Finance, SLU-P, and ISC sessions can warm start with control blocks created from an original descriptor, even when that descriptor is changed or deleted after the control blocks were created.

Logging onto ETO terminals

You can log on to your terminal in three ways.

You can use:

- The IMS **/OPNDST** command, and optionally include signon data.
- The SNA commands **INITSELF**, **INITOTHER**, or **USS LOGON**, and optionally include user data for signon.
- ETO autologon, which includes user data based on user descriptors or exit routines. The user data is then passed to signon.

If you specify ETO=Y, you can establish sessions with ETO terminals, but only if each of the following is true:

- An available logon descriptor provides sufficient information to create the control blocks necessary to accept the session request.
- In addition to the required logon descriptor, a 3270 or SLU–2 terminal requires an entry in the MFS device characteristics table that matches its screen-size and features. However, IMS does not search the MFS device characteristics table if the 3270 or SLU–2 terminal is identified as model 1 or 2.

- If the screen-size control byte is X'7F' in the PSERVIC field of the VTAM MODEENT macro, IMS searches the MFS device characteristics table using the alternate screen-size. If no match is found, IMS searches the MFS device characteristics table using the default screen-size. If you specify both sizes, and you want to use the default screen-size, the Logon exit routine (DFSLGNX0) must specify the default screen-size as an override.
- If the user cannot specify the logon descriptor in the user data, you must use the Logon exit routine (DFSLGNX0) or let IMS choose the default logon descriptor based on the terminal type.

Limiting dynamic logon to specific terminal types

If you want to limit dynamic logon to specific terminal types, delete the default logon descriptor for those terminal types you do not want to logon dynamically.

You can also reject the default logon for specific terminal types by using the Logon exit routine (DFSLGNX0). Dynamic logon for these terminal types fails.

Related concepts

"Criteria for selecting a default logon descriptor" on page 83
IMS provides a default logon descriptor for each VTAM terminal type. The logon descriptor name is based on the LU type and TS profile.

Creating and reusing LTERM control blocks

A user structure containing an LTERM can be created in a number of different ways.

A user structure containing an LTERM is created:

- When the user signs on, and the user structure does not currently exist
- When an asynchronous message is created for an LTERM, and the LTERM does not currently exist
- When you use the **/ASSIGN** command to assign an LTERM to a non-existent user.
- The **/ASSIGN** command is used to assign a non-existent LTERM to a user.
- The **/CHANGE USER** *user* AUTOLOGON command is directed to a non-existent user.

The user structure is allocated to a terminal after successful signon.

Using default CINIT or BIND user data formats

Each request for session initiation can include VTAM CINIT or BIND user data to provide logon descriptor or signon data. Your installation can provide a logon exit routine to process this data.

IMS can receive optional user data when you establish a session using one of the following methods: [8]

- Using an IMS **/OPNDST** command
- Using autologon
- The RTO provides a user logon

You can expand the user data formats to meet your own requirements. You can either supply the logon descriptor name in your logon exit routine by using user data, or you can create the logon descriptor name by using an IMS algorithm.

The user data appears in the CINIT user data field, and it is available to IMS when the VTAM Logon exit routine is scheduled. One optional parameter, the logon descriptor name, applies to the IMS logon process. The remaining parameters apply to the IMS signon process and, optionally, to RACF. During either process, IMS does minimum processing on the CINIT user data parameters before first calling the optional installation Logon exit routine (DFSLGNX0), and later calling the Signon exit routine (DFSSGNX0).

Although the Logon and Signon exit routines can translate any installation-defined user data format, IMS has defined a default user data format that:

[8] ETO ISC terminals do not have optional data available.

- The installation can expand.
- IMS can process in the absence of exit routines. This format is the logon descriptor name followed by signon data in the same format as the IMS **/SIGN** command.

Restriction: For warm session initiation of an STSN device, the user data must be the same as in the original logon.

Related reference

IMS control region exit routines (Exit Routines)

/SIGN command (Commands)

Format for CINIT user data parameters (System Programming APIs)

Signing on and queue LTERM allocation

Signing on to an ETO terminal identifies a user to IMS, creates a user structure, and connects this user structure to the terminal structure.

Users at VTAM terminals can sign on by:

- Issuing the **/SIGN** command.
- Signing on at the DFS3649 message screen.
- Providing user data with the session initiation request.

Users that establish a dynamic VTAM session with IMS must enter valid signon data before LTERMs are allocated to the session. You can require that the signon data be validated by a security product, such as RACF.

Providing signon data

Users can enter signon data by using one of several methods.

The methods that users can use to enter signon data include:

- Using the **/SIGN** command
- Including the signon data with the logon user data
- Coding the Logon exit routine (DFSLGNX0)

Providing signon data for ISC, SLU-P, Finance, and output-only devices

For dynamic ISC, SLU-P, Finance, and output-only devices, you must provide user signon data at session initiation.

IMS validates the signon data and allocates LTERMs to the terminals, based on:

User descriptors in IMS.PROCLIB
Applicable LTERM and user option defaults
Signon exit routine (DFSSGNX0) output

Restriction: A user cannot enter signon data from an output-only device during logon or when using the **/SIGN** command. For a dynamic output-only terminal, users must enter signon data at session initiation.

If a user forgets to include signon data during session initiation for an output-only device, IMS issues message DFS2085I (with return code 264) to the MTO. If signon data is omitted for dynamic ISC, SLU-P, or Finance terminals, IMS issues messages DFS3645I and DFS3672I.

Signing on multiple times

Users can concurrently sign on to one or more terminals. These terminals can be a combination of both static and dynamic terminals.

For both static and dynamic VTAM terminals, you must specify the EXEC parameter, SGN=M (multiple signons enabled).

For dynamic terminals, the name of the user structure that represents the user to IMS must be unique. You can use one of four methods to make the user structure unique:

- Use the Signon exit routine (DFSSGNX0) to return a 1- to 3-byte suffix to the user ID. The total length of the user ID plus the suffix cannot exceed eight characters.

  Recommendation: Use a naming convention that ensures unique user IDs. For example, you can create suffixed user IDs using the Signon exit routine (DFSSGNX0), and check the uniqueness of these user IDs by using IMS Callable Services.

  Recommendation: If you are operating in a sysplex environment, create a naming convention that ensures unique user IDs across the IMSplex by creating a suffix from the following:

  – As the first part of the suffix, the Signon exit routine can attach the unique part of the IMS system identifier, which is specified on the IMSID parameter, to the user ID.

  – As the second part of the suffix, use IMS Callable Services to choose an available value for the user on that IMS system.

  Example: A user with a user ID of USER signs on to IMSA. A unique suffix of A1 is created through this two-step process:

  1. The Signon exit routine appends the A from IMSA to the user ID.
  2. The Signon exit routine invokes IMS Callable Services. The exit routine then appends a 1, representing the first user named USER to sign on to IMSA.

  In this example, user IDs must be less than six characters in length.
- Use the Signon exit routine to specify that the user structure name is the same as the node name.
- Use the Signon exit routine to specify a name that is unrelated to the user ID or the node name.
- Specify a user descriptor name that is the same as the node name, causing the name of the user structure to be the same as the node name. You can do this by using one of the following methods:

  – Enter signon data that contains the user descriptor name.

  – Use the Signon exit routine to specify the node user descriptor.

  – Specify no user descriptor and let IMS use the node user descriptor by default.

Unless you specify SGN=M to enable multiple signons, the user ID for a dynamic user must be unique. If the user ID is used as the user-structure name, it must be unique, regardless of whether you specify SGN=M. If the user ID is not used as the user-structure name (for example, by using the Signon exit routine), you must specify SGN=M for multiple signons.

Recommendation: Assigning a single name to an IMS user is useful in determining output status. If you use multiple names for individual users (for example, when the Signon exit routine assigns them), you must provide a means to determine the user names that are created by signons.

Restriction: You cannot use the same name for a dynamic LTERM and a static LTERM that is defined during system definition.

Related reference

Signon exit routine (DFSSGNX0) (Exit Routines)

IMS callable services (Exit Routines)

Receiving DFS3649A, the signon required message

The ETO user must sign on before a session can enter transactions or commands.

The user can enter the **/RCLSDST** command only before signing on. IMS issues message DFS3649A (indicating that a **/SIGN** command is required) in each of the following situations:

- After a signon failure
- After a signoff
- After a logon in which no user data is entered

Any terminal user can then issue a **/SIGN ON** command to begin the next session.

You do not receive the DFS3649A message in any of the following situations:

- Signon data is included in the VTAM CINIT or BIND.
- The session is an ISC, SLU-P, Finance, or 3270 printer session.
- The terminal is a SLU-1 terminal running in unattended mode.
- The terminal is the subject of an autologon attempt.

Related reference
/SIGN command (Commands)
Related information
DFS3649A (Messages and Codes)

Receiving DFS3650I, the session status message

After a user successfully signs on (or if signon is not required), IMS issues message DFS3650I, indicating the status of the session with IMS.

Exceptions: In the following situations, IMS does not send message DFS3650I:

- When SLU-1 terminals run in unattended mode.
- When you specify the NOTERM option on the user descriptor.

Message DFS3650I provides information on session status, such as:

- Whether the user is in conversation mode
- Whether user output security exists for the terminal

Related information
DFS3650I (Messages and Codes)

ETO terminal-LTERM relationship

The system programmer is responsible for the relationship between a terminal that is in session with IMS and a particular user's LTERMs.

For a single-component terminal (such as a SLU 2), IMS creates a single default LTERM name that is the same as the user ID that was supplied during signon. Using an exit routine or a user descriptor, you can give the LTERM a different name.

For a multi-component terminal (such as a SLU P), you need to ensure that sufficient LTERMs are created in order to use that terminal. You can use exit routines, installation-created descriptors, or node user descriptors to ensure that a sufficient number of LTERMs is created. Otherwise, IMS creates a single LTERM that is allocated to the first component of the terminal. Regarding node user descriptors, the user needs to use the node user descriptor to establish the correct relationship between the LTERM and the node, either explicitly or in the Signon exit routine (DFSSGNX0).

It is possible for an application program to associate a specific LTERM name with a specific terminal. The system programmer must ensure that names of alternate PCBs are consistent with LTERM names defined in:

User descriptors
Signon exit routine
Insert exit routine
Recovery requirements for ISC, SLU-P, and Finance sessions

How IMS determines which queues to allocate

IMS uses the following method to determine which LTERMs to allocate.

1. If the user control blocks already exist within IMS, these control blocks are reallocated, and the Signon exit routine is called, if appropriate.
2. If the control blocks do not exist and a user descriptor is specified at signon, IMS looks for the specified descriptor.
   - If the descriptor specified is not the user ID, node name, or DFSUSER descriptor, IMS sends error message DFS3649A (with return code 148).
   - If the descriptor specified is a valid user descriptor for the user that is signing on, IMS builds a table of available descriptors and calls the Signon exit routine, if appropriate.
   - If the Signon exit routine exists and RC=0, IMS continues normal signon processing. If the return code does not equal 0, the signon is rejected.
3. If no user descriptor is specified at signon and no user descriptor is specified in the Signon exit routine, IMS chooses a valid descriptor from the following (in order):
   a. User ID
   b. Node name
   c. DFSUSER

Related reference
Signon exit routine (DFSSGNX0) (Exit Routines)

Setting special processing modes

After user signon, you can set the following processing modes by using IMS commands.

Exclusive mode
/EXCLUSIVE command

Preset destination mode
/SET command

MFS test mode
/TEST MFS command

Test mode
/TEST command

These special processing modes, except for the test mode and the preset destination mode, are retained for the user after signing off and are reestablished with the next terminal on which the user signs on.

When you issue the following commands, IMS creates the required control blocks for a terminal or user, or it maintains a user's status:

/EXC USER
Places a user structure in exclusive mode.

/STOP NODE or /STOP USER
Stops a node or user structure.

/TEST MFS USER
Places a user structure in MFSTEST mode. When a user signs on, IMS places the terminal in MFSTEST mode, if the terminal supports MFS.

/TRACE NODE
Traces a logged-on node.

Control blocks are not immediately deleted when special status is removed. They are deleted at the next checkpoint if they meet all deletion requirements. If an outstanding status exists, they are not eligible for deletion.

To reset terminal status and make the control blocks eligible for deletion at the next simple checkpoint, use the following commands:

/END
Clears exclusive and test modes.

/RESET
Removes the preset destination.

/RSTART
Starts stopped resources.

/START
Starts stopped resources. For other status reset by the **/START** command, see *IMS Version 15 Commands, Volume 2: IMS Commands N-V*.

/TRACE
Sets trace off.

Use the preceding set of commands if the control blocks exist solely for status retention.

You can use the Signoff exit routine, DFSSGFX0, to reset these states.

Related concepts

"Improving performance by deleting ETO control blocks" on page 103
With ETO, IMS can dynamically delete control blocks. Dynamically deleting control blocks reduces storage usage and can improve performance.

Printers with ETO

Two methods of implementing printer support exist in an ETO environment: direct printing and associated printing.

How you implement these two methods depends on how your application programs identify printer LTERM names.

Direct printing

Direct printing is a printing technique by which application programs insert messages to the VTAM LU name.

The dynamic LTERM and the user and terminal resources are all named after the VTAM LU name. Many application programs can queue data to the same printer LTERM, but this can create data interleaving problems.

Associated printing

Associated printing is a technique for directing application program printer output to a specific printer node name.

For associated printing, application programs insert messages to the queue that is associated with the screen-user queue name. User printing requests cause application programs to queue data to different printer LTERMs. This avoids data interleaving problems.

Printer signoff and signon are required to change the user queue. Overhead can be high if the printer has many users.

Associated printers are logged on automatically when their LTERMs have queued messages, usually during an IMS restart or during the creation of an LTERM.

Recommendation: Carefully plan how your printers are to be shared among application programs.

By implementing exit routines and application programs, the terminal operator can provide the destination during logon or signon.

Identifying printer node names

You can identify printer node names in one of two ways.

The ways that you can identify printer node names include:

- As logon user data, you can include one or more printer node names when a user establishes a session.
- Your installation can modify the MFS format for the DFS3649A greeting message in order to allow the user to supply the printer node names at signon time. During the signon, however, the Signon exit routine must be able to detect the printer node names as user data.

Coding the Signon exit routine for associated printing

To use associated printing, code the Signon exit routine (DFSSGNX0).

Code the Signon exit routine to do each of the following steps:

- Identify the printer LU name and user name for each screen user.
- Determine printer node names from the input user data.
- Name user-related LTERM structures to service the selected printers.
- Enable application programs to determine (using the user ID) the queues associated with a particular user so that the programs can insert to the correct message queue (alternate PCB).
- Pass the printer node names to IMS as associated print parameters.
- Determine the user name that is allocated to each printer when users supply printer node names. You can use either an algorithm or a table to make this determination. The exit routine should pass these user names to IMS, which then creates the necessary user control blocks.

  A unique name should exist for the user ID that is signed on and for each printer-related user that is required.

  Example: To indicate the name of a user's printer identification, add a "P" to the end of the user ID. If the user ID is AAA, the name of this user's printer identification is AAAP.
- Specify one value for each of up to four printers (the number of simple checkpoints before the user structure is deleted).

The application program can use the same method as the Signon exit routine in order to determine the printer LTERM name from the input terminal user ID and queue output data as required through an alternate PCB. When output data is queued, IMS allocates the user structure to the correct printer and delivers the output.

After associated printer LTERMs are allocated and then emptied, the queues are deallocated from the terminal.

Related reference

Signon exit routine (DFSSGNX0) (Exit Routines)

Defining your printers

Printers are usually output-only devices; however, they can be implemented so that they send input.

You need to decide whether to have single-user or multiple-user structures share the same printer. If your application programs are sensitive to queue names, you might be limited to one or the other approach.

Single-user structure

Using a single-user structure that represents a printer is the simplest method for defining a printer. In this case, messages sent to the printer are printed in the sequence in which they originate. Interleaving of output can occur. Multi-segment messages always have all segments printed in contiguous sequence. Multiple messages might behave differently, depending on their method of origination.

Multiple-user structures

Multiple users are supported for printers. All messages for a single user are printed, and the next user is selected for printing only when the current user has no more output. Although interleaving of messages for the same user occurs as it does for single-user structures, the switching of users can also be equivalent to message interleaving from an application standpoint. Application and operational awareness of the way printers are shared is important.

The challenges of sharing printers are not unique to dynamic terminals, and in fact are the same as for static terminals.

IMS prints a separator page whenever the next message to be printed is from a different user. You can control the contents of this separator page by using an exit routine to change its content; however, this page is always printed, even if blank. Messages from the same user are not separated by a separator page. Using the NOTERM option can prevent the DFS3650 separator message from being used.

Sharing printers using ETO

Several users can share printers by using the same output terminal.

Two methods exist for using the same output terminal.

- Users can define a single-user descriptor that contains all the LTERM names that are used to deliver data to an output device (node).
- Users can define autologon parameters for a node.

Before deciding which of these methods to use, ask these questions:

- Are the LU-to-LTERM relationships generally unchanging?
- Is interleaving at the message level acceptable?
- Is delaying one user's output acceptable while another user's output is being printed?
- Is continuous autosignoff and autologon processing overhead too excessive for a particular terminal because of the minimal message delivery rate of each LU user?

The answers to these questions can help to determine your method of implementation:

- If the answer to all of the questions is "yes", consider creating a single multi-LTERM user for the printer.
- If any of the answers is "no", consider dynamic allocation of multiple-user LTERMs, using autologon.
- You can implement a combination of these two options.

Operator commands

This topic describes using the **/OPNDST** and **/ASSIGN** commands as they are used with ETO.

/OPNDST

The VTAM mode table that is used when the terminal first logs on determines the device characteristics. However, if the first reference to a terminal is through an **/OPNDST** command, the MODETABLE operand of that command determines the device type.

Omitting the MODETABLE operand on the command causes it to default, which might not be desirable. After the terminal is in session with IMS, the rules for block deletion apply. The device type remains set until the block is deleted; if the terminal is closed and then reopened, IMS uses the existing block, if available. If a block is deleted, that block is rebuilt during the next terminal logon. As a result, the **/OPNDST** command can have different results, depending on two things:

- Whether the block is still available (and has not been deleted).
- Whether the exit routine or descriptor has been changed since the previous initialization.

/ASSIGN

You can use the **/ASSIGN** command to move queues from one terminal to another. You cannot use it to reassign a static terminal to a dynamic (ETO) terminal, or to assign a dynamic terminal to a static terminal.

Related reference

/ASSIGN command (Commands)

/OPNDST command (Commands)

System definition parameters for ETO

This topic describes the system definition parameters you can use with ETO.

Setting DEADQ status time with the DLQT parameter

User control block structures are normally created in several situations.

The situations in which user control block structures are created include:

- When a terminal is logged on and a user signs on.
- When the AO exit routine (DFSAEOU0) inserts a message to an LTERM or transaction.
- When an asynchronous transaction output message is sent, or a terminal message switch or **/BROADCAST LTERM** command is issued.

Messages might be sent to destinations that are unknown, no longer valid, or nonexistent. IMS sets a status of dead-letter queue (DEADQ) when an ETO user control block structure or its associated message queues have not been accessed within the time limit that is set by the DLQT execution parameter. (For DLQT, although valid values are 1-365 days, a value of 1 is not usually recommended. It can result in many premature and misleading DEADQ status settings during the first checkpoint.) The DEADQ status can be set regardless of whether messages have been queued for the user or whether the user is still allocated to a terminal. The DEADQ status is set during IMS checkpoints, and the MTO is notified with message DFS3643.

If the user has messages queued, remove the DEADQ status by signing on the user or by issuing the **/DEQUEUE** or **/ASSIGN** command.

User control block structures without queued messages can result in a DEADQ status. User control blocks are not deleted if a special status is pending. The status might have been set during the prior signon (such as response or conversation mode) or as a result of a command (such as **/STOP** or **/EXCLUSIVE**). If the control block remains unused for longer than the time specified for the DLQT execution parameter, IMS assigns the control block a DEADQ status, and the MTO is notified with message DFS3643.

If the user control block has DEADQ status, the status is removed when the user signs on or during the next checkpoint after all messages are dequeued and recoverable status conditions are removed using appropriate commands. In the latter case, the control blocks for the user and associated message queues are also deleted.

User control block structures that have DEADQ status might or might not be allocated to a terminal. In the case of LU type–6, SLU-P, and Finance terminals, the user structure can be allocated to the terminal with no active session. Logoff or other session termination can leave these terminals pending message recovery (SNA STSN). The user remains allocated to the terminal, and messages might or might not be queued. (This combination is not possible for other VTAM terminal types, because a deallocation of the terminal and user ID is forced at logoff and signoff.)

If the user control block structure is allocated to an LU type–6, SLU-P, or Finance terminal, remove the DEADQ status by logging on and signing on the user, or by using the **/DEQUEUE** command. For LU type–6 (ISC) terminals use the **/DEQUEUE** command, or force the LU type–6 session to cold start. A forced-session cold start is a **/STO NODE**, **/ASSIGN (USER TO VTAMPOOL)**, **/STA NODE** sequence that is valid only for LU 6. Forced cold start is not possible for SLU-P or Finance terminals. Clearing the allocation of an ETO user to a SLU-P or Finance terminal requires an IMS cold start.

In a shared-queues environment, you can use the **/DISPLAY QCNT MSGAGE** command to find the age of a queue.

Autosignoff (ASOT)

Autosignoff deallocates users from an idle session. If no activity occurs on a session within an allotted time, users are automatically signed off and must sign on again in order to use the session.

The system programmer sets the autosignoff time using the ASOT parameter in the DFSPByyy member. The time value on the ASOT EXEC parameter does not apply to 3600, SLU P, or ISC devices.

If more than one allotted time value exists, IMS uses the following criteria to determine which allotted time value to use:

- If the valid ASOT value is specified in the user descriptor, this allotted time value is used.
- If the user descriptor does not specify an allotted time value, the allotted time value from the logon descriptor is used.
- If the logon descriptor does not specify an allotted time value, the time value from the DFSPByyy member is used.
- If the DFSPByyy member does not specify an allotted time value, the default value of 1440 is used.
- If the value on the DFSPByyy member is not valid, the default value of 10 is used.

The Logon exit routine (DFSLGNX0) can override the ASOT and ALOT values during logon. The Signon exit routine (DFSSGNX0) can override the ASOT value during signon, even when the control block structure exists.

The values for the allotted time specified on the ASOT parameter in the DFSPByyy member are:

- `ASOT = 0`
  - The user is signed-off immediately when no output is available to be sent. This specification is normally used with Autologon terminals for signoff immediately when:
    - No IMS input or output message is available
    - After the last available output message completes
  - This specification is not recommended for interactive terminals such as 3270 or SLU2 terminals. These terminals sessions normally return a PA key to continue following signon. Idle time results in immediate signoff, not waiting for terminal input.

  The value on the DFSPByyy member is not used for these device types.
- `ASOT = (10 - 1439)`

  The user is signed off after the allotted number of minutes has elapsed without terminal activity.
- `ASOT = 1440`

  The user is never automatically signed off. This is equivalent to not having autosignoff. The system default value for SLU-P, 3600/Finance, and ISC terminals is 1440.

After autosignoff completes, IMS attempts to locate a user that has the same node name that is waiting for autologon. If IMS finds another user with output waiting, the user is allocated to the terminal, and the queues are drained.

Autologoff (ALOT)

Autologoff can terminate a session with IMS for a terminal that has been signed off for an allotted period. After an allotted time, the terminal is automatically logged off.

The system programmer sets this time on any of the following:

- On the ALOT (auto logoff time) parameter in the DFSPByyy member
- On the logon descriptor that is used to create the session control blocks
- On the EXEC parameter at initialization

If more than one allotted time value exists, the following criteria are used to determine which allotted time value to use:

- If the ALOT value is specified on the logon descriptor, this allotted time value is used.
- If the logon descriptor does not specify a valid allotted time value, the time value specified in the DFSPByyy member is used.
- If the DFSPByyy member does not specify an allotted time value, the default value of 1440 is used.
- If the value on the DFSPByyy member is not valid, the default value of 10 is used.

You can specify ALOT=1440 on the logon descriptor for a 3600/Finance, SLU-P, or ISC terminal in order to specify that no autologoff is desired. The system default value for these devices is 1440. Logon descriptors created for ISC terminals should have ALOT=1440 specified to show that autologoff should not occur. The Logon exit routine (DFSLGNX0) can override the ALOT value during logon, even when the control block structure exists.

The values for the allotted time specified on the ALOT parameter in the DFSPByyy member are:

ALOT = 0
The terminal is logged-off immediately when no signon is in effect. This specification is normally used in terminal sessions when the user is signed-on automatically during the logon process. During autologon, signon data can be provided in one of the following ways:

- Signon data supplied by the **IMS /OPNDST** command
- Signon data supplied by logon user data (BIND)
- Signon data supplied by logon exit (DFSLGNX0)

There are two modes of operation for using ALOT=0, either of which can be set using the DFSINTX0 User Initialization Exit parameter list.

In default mode, when signon errors are encountered, the session is automatically signed off and then logged off; no message is sent. If you do not supply the DFSINTX0 exit, or you supply the exit and indicate default mode for ALOT=0, then signon data must be supplied during the logon process. All of the following error conditions result in automatic logoff:

1. A non-signon, or errors detected during signon or input processing, result in immediate logoff.
2. **/SIGNOFF** results in immediate logoff.
3. **/SIGNON** signs off the current user and signs on a new user. However, errors encountered during the signon process, such as detection of an incorrect or expired password, result in immediate logoff.

Restriction: Default mode should not be used for interactive terminal sessions that require a response to the DFS3649 message; these sessions will not wait for input signon and will logoff immediately.

In alternate mode, when signon errors are encountered, the session is automatically signed off, a message is sent and the session is logged off. Signon data can be supplied but is not required. All of the following error conditions result in automatic logoff:

1. A non-signon error detected during input processing results in immediate logoff.
2. No signon data has been provided by the logon serrated (BIND) or the Logon Exit (DFSLGNX0).
3. A **/SIGNOFF**, or errors resulting from a **/SIGNON**, cause message DFS3649(A) (Signon Required) to be sent, and a fixed ten-minute timer set to wait for a new signon. If no signon occurs during that interval, then the session is logged off.

ALOT = (10 - 1439)
The session is terminated after the allotted number of minutes has elapsed without a signed-on user.

ALOT = 1440
The session is never automatically terminated. This is equivalent to not having autologoff.

Autosignoff and autologoff timer

The VTAM I/O Timeout Facility (if active) detects users or sessions that should be automatically terminated.

A timer pops at intervals of one minute if the VTAM I/O Timeout Facility is active, or five minutes if the Timeout Facility is not active. The timer starts a routine that determines which resources are due for autosignoff or autologoff. The specified timeout value is the minimum value for which a user or session is automatically terminated. Termination actually occurs the next time the timer pops after the specified timeout value.

Autologon

Instead of using the SHARE option on the TERMINAL macro to request that IMS automatically initiate a terminal session when output is available, ETO offers autologon support for ETO terminals and users. You specify autologon parameters when defining the user to IMS.

Definition: *Autologon* allows IMS to log on and sign on your terminal automatically. If you specify the autologon option for a user, the queuing of data to any of the user queues causes IMS to establish a session. You can specify autologon using:

- AUTLGN= parameter on the user descriptor
- Destination Creation exit routine (DFSINSX0)
- Signon exit routine (DFSSGNX0) for associated printers
- **/CHANGE** command with the AUTOLOGON keyword

Autologon includes both automatic logon and automatic signon. At restart, IMS attempts to start sessions with those terminals that are defined with autologon and that have queued data waiting. After the session is established, IMS automatically signs on the terminal. If a queue of waiting users exists and the allocated queues are drained, the autologon user is signed off, regardless of an existing ASOT value.

IMS manages a serial queue of waiting users if more than one user is contending for the same autologon terminal. After the autologon user is signed off, the next autologon user for the same terminal is automatically signed on. When all autologon users have signed off, the terminal is free to begin its ALOT cycle in order to terminate the session.

Autologon replaces the TERMINAL macro OPTIONS=SHARE for static terminals. OPTIONS=NOASR (no automatic session restart) on the logon descriptor is ignored for autologon printers. IMS always assumes OPTIONS=ASR for autologon printers.

Autologon is normally specified for output-only terminals. Autologon and occasional users generally do not share the same terminal session, but sharing terminal sessions is possible for interactive terminals. Interactive users on terminals that are subject to autologon must supply user signon data with the session initiation request (logon) in order to avoid contention with autologon output. Occasional terminal users can use autologon in order to have output delivered to default terminals when the output becomes available after signoff. If a terminal is stopped, the **/START NODE** command does not start a session. The **/OPNDST NODE USER** command can be used to restart the session.

Assigning output

The following topics discuss managing asynchronous output and output destinations.

Asynchronous output

Asynchronous output can easily be sent to an invalid destination by a simple typographical error.

This is because ETO provides IMS the flexibility to create user structures for any authorized user that is signed on. IMS automatically creates user structures for message switches and for the application insert call (ISRT) process when the LTERM cannot be found.

With ETO, all destinations are valid unless they are rejected by exit routines. Therefore, you can mistakenly create queues for users if you make typographical errors when entering any of the following:

Alternate PCBs
Message switches
/BR0ADCAST commands
MSC output

The LTERMs used in the previous situations are known as *dead-letter queues*. IMS provides commands for the MTO to monitor these queues and dispose of them.

The two types of asynchronous output destinations are valid and invalid.

A *valid destination* is one intended to receive output. An *invalid destination* is one that is not intended to receive output (for example, a misspelled destination).

Asynchronous output to a valid destination

You can send asynchronous output (such as inserts, broadcasts, and message switches) only after enabling ETO and defining a valid ETO descriptor and a valid destination LTERM name.

ETO is enabled by specifying ETO=Y in the IMS or DCC startup procedure.

The ETO descriptor is used for creating a user structure.

If control blocks exist for a previously created user structure and LTERM, the control blocks are reused.

Asynchronous output to an invalid destination

IMS refers to data that cannot be delivered as "dead letter".

Data cannot be delivered in each of the following situations:

- No autologon destination is available for queued output.
- The user ID to which the data is associated is not a valid user ID.
- The user signon is always rejected by an installation Signon exit routine.
- An invalid destination is specified on the input or output message (for example, resulting from a typing error).

You can specify a DLQT value on the EXEC parameter at initialization in order to automatically notify the MTO when LTERM queues exceed this value and have not dequeued data or removed the status. You can use each of the following commands against dead-letter queues:

- Use the **/DISPLAY USER DEADQ** or **/DISPLAY STATUS USER** command to identify users whose LTERM queues are older than the dead-letter-queue time (DLQT), and who have not dequeued data or removed the status.
- Use the **/ASSIGN** command to reassign dead-letter queues to other dynamic users so they can review queued data.
- Use the **/DEQUEUE** command to purge data on the dead-letter queues.

In a shared-queues environment, you can use the **/DISPLAY QCNT MSGAGE** command to determine the messages that are considered to be dead-letter queues.

Related reference
IMS commands (Commands)

Delivering output messages to non-originating terminals

IMS sends your output to the terminal at which you are signed on.

Using ETO, you can receive your output messages at a different terminal than the one from which you entered the input. However, the input and output messages are formatted subject to the MFS specifications defined for both the terminal and messages, as follows:

- MFS formatting is mandatory for 3270R (non-SNA) and SLU-2 terminals, except when you use MFS bypass. MFS formatting is optional for all other VTAM terminals. Use MFS on a message-by-message basis, based on MID and MOD control block availability. MFS paging support is not available for SLU–1 or NTO terminals.
- You must define MID and MOD control blocks by using device statements that generate appropriate DIF and DOF control blocks. This allows you to map the message to and from a specific terminal type at the time that the message is sent to or received from the terminal. Default mapping occurs when the appropriate MFS blocks are not available. IMS issues error message DFS057 when the format blocks cannot be found.

If you plan to deliver output messages to non-originating terminals, you must develop or expand MFS formats, procedures, and restrictions. This allows users to move freely between terminals.

Inadvertent output data streams

Using ETO, dynamic terminal users can move freely between terminals; limited only by installation constraints. It is possible to erroneously send terminal-specific data to the wrong terminal type by mistyping the IMS **/ASSIGN** command.

When data is sent to the wrong terminal, the data, when delivered, has errors or is not recognizable. Be sure to create MFS definitions for all terminal types for which users can log on.

Signing off

Signing off of an ETO terminal ends the identification of a user to IMS, and in most cases disconnects the user structure from the terminal structure and deletes the user structure.

When a user signs off from an ETO VTAM terminal, IMS calls the Signoff exit routine (DFSSGFX0).

Recommendation: If you have provided a Signon exit routine (DFSSGNX0) that maintains system information, provide a Signoff exit routine (DFSSGFX0) also, to complement that processing.

Logging off

When a user logs off from a VTAM terminal when ETO is used, IMS calls the Logoff exit routine (DFSLGFX0).

Recommendation: If you have provided a Logon exit routine (DFSLGNX0) that maintains system information, provide a Logoff exit routine (DFSLGFX0) also, to complement that processing. Ensure that the Logoff exit routine handles all non-MSC and non-LU 6.2 terminals with which IMS communicates.

Using the Logoff exit routine, you might want to maintain a count of the terminals that are logged on.

Improving performance by deleting ETO control blocks

With ETO, IMS can dynamically delete control blocks. Dynamically deleting control blocks reduces storage usage and can improve performance.

If special terminal processing options (TRACE and STOPPED) are reset, IMS deletes session control blocks if one of the following occurs:

- No user is signed on, and a checkpoint occurs.
- The session is terminated normally or abnormally by either an MTO command or by an autologoff timeout.

If the node is a 3600/Finance or SLU P terminal, message resynchronization is necessary. The control blocks are not deleted following warm-session termination, but will be deleted following a `/CHANGE NODE COLDSESS` command. For ISC terminals, the control blocks are deleted after a cold-session termination. The control blocks remain, however, after an ISC warm-session termination.

If the user resets special processing options, such as those that exist after issuing a /SET, /TEST MFS, or /EXCLUSIVE command, and issues the /SIGN OFF command (or is automatically signed off), IMS deletes the user control blocks if:

- No messages are queued to any LTERMs related to this user.
- The user is not in conversation, Fast Path mode, or full-function response mode.

If the preceding conditions exist, dynamically created user control blocks are deleted. If the preceding conditions do not exist, the user control blocks can continue to exist until the conditions exist or until an IMS cold start occurs. After special processing options are reset and if all other criteria for deletion exist, the control blocks are deleted at the next checkpoint.

Important: User control blocks can be saved across session and IMS restarts by using the /CHANGE or the /ASSIGN commands with the SAVE keyword. These user control blocks are then retained until the commands are reentered with the NOSAVE keyword.

For terminals or users in full-function response mode, IMS does not delete user control blocks after a terminal logoff or a user signoff if both SRMDEF=LOCAL and RCVYRESP=YES, because in this case the full-function response mode is recoverable.

Note: In an IMSplex, if the status recovery mode is GLOBAL or NONE, the local control blocks are deleted immediately after logoff or signoff.

IDC0 Trace facility

You can use the IDC0 Trace facility to diagnose logon and logoff errors.

This facility provides information that the IMS message DFS3672I cannot provide. To use the facility, enter: **/TRACE SET ON TABLE IDC0**. The facility traces the following events:

- Errors that occur in the IMS VTAM exit routines (within module DFSCNXA0). These errors are also identified in a DFS3672I message, regardless of whether the IDC0 Trace is in effect.
- Errors that occur when attempting to log onto a nonexistent VTAM node (such as entering a **/OPNDST** command for a nonexistent terminal). IMS issues associated messages DFS2061I or DFS2062I.
- Synchronization anomalies that occur between the time that an IMS VTAM exit routine completes processing and the time that the request is accepted by normal IMS processing. The result is a X'6701' log record identified with a VTPO string.

Related reference

/TRACE TABLE command (Commands)

Format of the 6701 log record with VTPO identifier (Diagnosis)

IDC0 trace table entries (Diagnosis)

ETO and LU 6.1 (ISC) terminals

For LU 6.1 (ISC) terminals, IMS supports parallel sessions to the same node name. In this case, a separate structure is built for each session. However, each session and its associated structure operate independently, as a separate terminal.

ISC supports an SNA-defined user-data area within the BIND. When establishing a session for ISC, each half-session partner is identified through an appropriate session qualifier that is included as user data with the logon. These two qualifiers cannot be specified using the DFSLGNX0 exit routine. One belongs to each half session. IMS uses the session qualifier of the current half session as a user structure. This user structure is used to allocate an associated set of LTERM queues and to automatically provide a RACF signon, if required. The other half-session qualifier is saved with the IMS user structure. Both qualifiers are used for session-restart requests and SNA STSN message resynchronization after session failures.

Restriction: The SNA-predefined format for user data does not support some of the parameters and options of the non-ISC end-user format:

- The RACF password and group name are not supported. IMS supports RACF signon for ISC with PASSCHK=NO during user structure allocation as part of session initiation.
- The LOGOND and USERD are not supported. IMS uses defaults, unless specified on the Logon and Signon exit routines.
- When autologon is generated for ISC terminals, the AUTLDESC keyword in the user descriptor is ignored, and the LOGOND keyword in the user data is omitted.

Related tasks

"Using default CINIT or BIND user data formats" on page 90
Each request for session initiation can include VTAM CINIT or BIND user data to provide logon descriptor or signon data. Your installation can provide a logon exit routine to process this data.

Related reference

Logon exit routine (DFSLGNX0) (Exit Routines)

ETO and STSN terminals

This topic provides information on administering ETO for STSN terminals.

SNA STSN terminal considerations

ETO terminals that use the SNA STSN function (Finance, SLU-P, and ISC) must provide a user queue name during logon, because this queue is used to resolve the sequence number exchange that is part of the IMS connection process for this type of terminal.

The ways to provide the user name include:

- CINIT data sent by the terminal
- CINIT data provided using a separate terminal host product, such as VTAM unsolicited system services
- A user Logon exit routine (DFSLGNX0), except for ISC
- An IMS command (**/OPNDST** can specify user name)
- Autologon parameters supplied by user descriptors or by the Destination Creation exit routine

The Logon exit routine is the last opportunity to provide the user name. If no user name is provided for ETO STSN terminals, the logon is rejected with an error message. This differs slightly from other terminal types, which allow user signon after the logon has occurred. A signon is required before being able to use the terminal for IMS activity (transactions or commands), so the difference is only in requiring the signon data earlier for STSN terminals. The **/SIGN** command is supported for STSN terminals.

ETO and 3600/Finance and SLU P

You can sign on to static system-defined 3600/Finance and SLU P terminals in one of two ways: using the **/SIGN** command or using logon user data.

You can change the signon identification by using another IMS **/SIGN** command at any time. LTERMs are assigned to the terminal during system definition and are not affected by the signon process or by the SNA STSN message recovery process. The signon simply provides user access and input authorization.

IMS supports 3600/Finance and SLU-P terminals as dynamic terminals. They can use autologon or signon data with the logon request in order to dynamically allocate user structures. Signon data must be provided at session initiation for ETO 3600/Finance and SLU-P terminals. Signon data can be supplied in the Logon exit routine (DFSLGNX0) at the cold start of a session. The LTERMs and user IDs allocated at the cold start session are retained across sessions and IMS outages because of the VTAM STSN message resynchronization requirements. This requires that the same user signon data be used for subsequent warm-start sessions to both reverify the user and to allow message resynchronization.

When dynamic XRF Finance and SLU-P terminals are defined as XRF class–2, automatic re-signon and logon occur at takeover time.

/SIGN support for ETO STSN devices: ISC, Finance, and SLU P

For ETO STSN devices, user data is required at the time that the session is allocated to create the user structure.

After the user structures are created and allocated to the terminal, **/SIGN** commands are accepted from ETO STSN terminals.

When you issue the **/SIGN** command from an ETO STSN terminal, IMS initiates a complete signon process to create the security profile associated with the session for the new user.

When the user signs off, the user's security profile is deleted, leaving the session without any security. RACF rejects all access to RACF-protected resources. The DFS3662 message is displayed if the failing resource is a command. The DFS2469 message is displayed if the failing resource is a transaction. When a user signs on to an STSN device, the same user structure allocated during session allocation is used for the new user. IMS updates the security profile of the session and stores the signon information.

The user ID of a user that is signing on to an ETO Finance, SLU-P, or ISC device with a **/SIGN** command is a different name from that of the user structure name. Such users do not require suffixing by the Signon exit routine (DFSSGNX0) in order to support multiple signons, because the user structure for these devices was already created during the session initiation.

Restriction: Because the user structure allocated to a Finance, SLU-P, or ISC device cannot be changed, most of the options available to the Signon exit routine are not supported, and the work areas are not passed to the Signon exit routine.

The DFS3650 message is displayed after a signon, and the DFS058 message is displayed after signoff. The user field in the DFS3650 message reflects the user structure's name rather than the RACF user ID. This is the same as for any other ETO terminal.

Restriction: Issuing the **/SIGN** command from STSN output-only devices is not allowed.

Related concepts

Specifying IMS execution parameters (System Definition)

Exit routines (Exit Routines)

Conversation mode and response mode with ETO

ETO user structures are distinct from static terminal structures, because terminal status is maintained for each user rather than for each terminal.

An IMS conversation at a static terminal is distinct from an IMS conversation at a dynamic terminal, even for the same user name. The conversation at a static terminal can be held, and must be released (resumed) at the static terminal. It cannot be moved to a dynamic terminal.

With ETO, an IMS conversation can be resumed at the same terminal or another dynamic terminal. Because the conversation is an attribute of the user structure, it normally follows the user to a different terminal when the user signs on to IMS. The following of the conversation attribute can be a cause of confusion, especially if the user is not aware of which terminals are static and which are dynamic. Also, if an exit routine selects different user structures for signons to different physical terminals, confusion can occur.

Resume full-function and Fast Path response mode at the same static terminal. For ETO, resume full-function and Fast Path response mode from the same or another dynamic terminal. This ability to resume the Fast Path response mode from any dynamic terminal is similar to the situation with conversations described in the previous paragraph, and the same considerations apply.

Conversation mode

For ETO, conversations are associated with the user—not the terminal that initiates the conversation. Conversations are also associated with the terminal, but only while the user is signed on. By signing off, the user can continue a conversation on a different terminal. This flexibility requires the installation to address output formatting problems.

Users in conversations that are not in response mode can sign off. Regardless of response mode, users in conversation can be automatically signed off through autosignoff or by using an MTO command. Any form of signoff leaves the terminal available for the next user. The conversation mode status follows the user to the next terminal or, with the Resource Manager and global status recovery mode (SRM=GLOBAL), on a different IMS in the IMSplex.

Response mode

Response mode is defined on the TERMINAL macro for static terminals, on the ETO user descriptor for dynamic (ETO) users, and on the TRANSACT macro for transactions. Also, response mode is either full function or Fast Path. You can also use response mode with conversation mode, if you are not running Fast Path. Response mode is primarily associated with the user and the transaction, rather than with the dynamic terminal.

When a user is in response mode, the keyboard or input response is locked until the output reply is available. During this time, it is not possible to enter input at the terminal. Normal signoff and logoff commands are not allowed. However, these functions can occur automatically during abnormal session termination. This can happen in one of three ways:

- VTAM can detect an error and end abnormally (abend).
- The MTO can issue the IMS `/CLSDST` or `/STOP` command.
- IMS can autosignoff after the specified autosignoff interval.

Regardless of how signoff occurs, if RCVYRESP=YES or RCVYFP=YES, the response mode is retained for the user that has been automatically signed off. The user's response mode operation is re-established with the next terminal on which the user signs onto and remains until the response-mode output reply is available.

The master terminal operator can reset the Fast Path response mode of an ETO dynamic user before a response is returned by issuing the `/STOP USER` and `/START USER` commands in sequence from the master terminal. The master terminal operator can also reset the Fast Path input response mode of a static node by issuing the `/STOP NODE`, `/START NODE` commands in sequence from the master terminal.

Related concepts

“Delivering output messages to non-originating terminals” on page 102
IMS sends your output to the terminal at which you are signed on.

Part 4. External subsystem attach facilities

IMS provides several options for accessing external subsystems from an IMS system.

© Copyright IBM Corp. 1974, 2020

Chapter 6. DB2 Attach Facility

Java message processing programs (JMPs) and Java batch programs (JBPs) in IMS can access Db2 for z/OS data using the DB2® Resource Recovery Services Attach Facility, referred to here as the DB2 Attach Facility.

Each dependent region that is set up for this support builds its own RRSAF thread to access Db2 for z/OS data. This thread enables the coordination of updates that the application program makes with the resources of both IMS and Db2 for z/OS resource managers. When IMS JMPs and JBPs use the DB2 Attach Facility to access Db2 for z/OS data, IMS is not the sync point coordinator of updates and commits, as it is with ESAF. With the DB2 Attach Facility, IMS is a participant, and z/OS Resource Recovery Services is the sync point coordinator.

Preparing your system to use the DB2 Attach Facility

To prepare your system to use the DB2 Attach Facility, you must perform two tasks.

To use the DB2 Attach Facility:

1. Add the attachment facility definition to the IMS PROCLIB data set.

   Use of the DB2 Attach Facility requires that a subsystem member (SSM) be defined in IMS.PROCLIB. If an SSM member does not already exist in IMS.PROCLIB, you must create one. The SSM contains an entry for the Db2 for z/OS system with which IMS and the application program communicate. All Java dependent regions in IMS access a single Db2 for z/OS system.
2. Make the Db2 for z/OS RESLIB available to the IMS JMP and IMS JBP regions.

   After defining the attachment facility, you must provide the Java regions in IMS with access to the Db2 for z/OS RESLIB. In the JCL for the JBP and JMP regions types, add the Db2 for z/OS library definition using the DFSDB2AF DD statement, which points to the Db2 for z/OS libraries that contain modules used by RRSAF. The Db2 for z/OS libraries must be APF-authorized.

Related concepts

Accessing external subsystem data (System Definition)

Managing how your Java dependent regions access Db2 for z/OS

Java application programs running in IMS dependent regions can access Db2 for z/OS under syncpoint control of z/OS Resource Recovery Services if a DB2 Attach Facility definition is included when the IMS control region is started.

The initialization processing of the IMS control region prepares for access to Db2 for z/OS. When Java dependent regions are subsequently started, application programs in those regions can make direct calls to both Db2 for z/OS and IMS.

Initialization of the DB2 Attach Facility does not affect the execution of other types of dependent regions. The DB2 Attach Facility definition can be retained in the IMS.PROCLIB member, even if Java dependent regions are not used. If a DB2 Attach Facility definition exists in the IMS.PROCLIB member, and the Db2 for z/OS library is defined in the Java dependent region JCL, all Java dependent regions that start will build an access thread to Db2 for z/OS. If the DB2 Attach Facility definition exists, but the Java dependent regions do not require access to Db2 for z/OS, you can prevent access threads from being built by stopping access to the Db2 for z/OS system. Use the **/STO SUBSYS** command, which stays in effect until a **/STA SUBSYS** command is subsequently issued.

Related reference

/START SUBSYS command (Commands)

/STOP SUBSYS command (Commands)

© Copyright IBM Corp. 1974, 2020

Chapter 7. External Subsystem Attach Facility (ESAF)

The External Subsystem Attach Facility enables BMP, IFP, JBP, JMP, and MPP application programs to access databases managed by other subsystems in addition to DL/I databases.

To enable access to the data resources of an external subsystem (ESS) product from IMS applications, the ESS must provide functions necessary for it to attach to the IMS subsystem and to, jointly with IMS, coordinate data access. The IMS Attach Facility presents a programming interface to the external subsystem product. Certain steps are required to install ESAF, which are described in the following information. Other IMS publications also contain ESAF information and references are included, where applicable.

Multiple external subsystems can only be attached by an online IMS system. These subsystems can be of the same, or of different, product types. The installation defines the external subsystems to IMS. A given IMS dependent region can have access to all external subsystems defined to the IMS system, to just a subset, or to none according to installation specifications. An application program executing in a dependent region can access more than one different subsystem. The installation defines a unique token for each subsystem, which IMS uses in routing application calls for external resources. Application program access to more than one subsystem of the same type is supported by IMS, but might not be supported by the external subsystem.

The facility provides for synchronization of external subsystem data resources with IMS data resources. For synchronization processing, IMS is the recovery coordinator and is responsible for directing commit or abort actions on behalf of its application programs. External subsystems are participants in the process and commit or abort data updates by IMS applications according to direction given by IMS. When resources are to be committed, IMS polls the participants as to whether or not they are ready to commit before giving final commit (or abort) direction.

You can also configure a Fast Database Recovery (FDBR) region to recover work on the external subsystem. When a FDBR region is monitoring an IMS system that fails, it receives information about indoubt work on the external subsystem from the ESAF indoubt notification exit routine (DFSFIDN0).

For IMS batch, IMS is allowed to attach only to one external subsystem. IMS expects this external subsystem to be the Recovery Coordinator. This external subsystem has no way of coordinating with any other external subsystem that IMS attaches to, so IMS is restricted to only one external subsystem attachment in batch.

The External Subsystem Attach Facility is different from the coordinator controller (CCTL) associated with DBCTL.

JMP and JBP regions can access control-region-defined Db2 for z/OS subsystems that use the COORD=RRS parameter in the IMS.PROCLIB member. If this connection method is chosen for the JMP or JBP region, you must add a DD statement (DFSDB2AF) in the DFSJMP or DFSJBP procedure that points to the Db2 for z/OS libraries.

This topic contains General-use Programming Interface information.

Related tasks

“DB2 Attach Facility ” on page 111
Java message processing programs (JMPs) and Java batch programs (JBPs) in IMS can access Db2 for z/OS data using the DB2® Resource Recovery Services Attach Facility, referred to here as the DB2 Attach Facility.

Accessing Db2 for z/OS databases from JMP or JBP applications (Application Programming)

© Copyright IBM Corp. 1974, 2020

What the external subsystem must provide

The External Subsystem must provide three things: the External Subsystem Attachment Package (ESAP), the External Subsystem Module Table (ESMT), and the Resource Translation Table (RTT).

External Subsystem Attachment Package (ESAP)

The IMS Attach Facility uses an exit routine interface. That is, to accomplish external subsystem access from dependent regions, IMS activates exit routines at certain processing points. These exit routines must be supplied by the external subsystem. The exit routine functions are prescribed by IMS; the external subsystem supplies its unique implementation. The exit routines must, in fact, provide the actual linkage to the external subsystem. IMS is not sensitive to the linkage mechanism used.

IMS loads external subsystem-supplied exit routine modules in the control region and in each dependent region that can access the external subsystem. The external subsystem can supply additional modules needed for attach exit routine processing; IMS loads these modules as well. The external subsystem modules provided for attach processing in the IMS regions make up what IMS calls the External Subsystem Attachment Package (ESAP).

External subsystem module table (ESMT)

The external subsystem must specify the modules that IMS is to load in an external subsystem module table (ESMT). The external subsystem creates the module table using macros provided by IMS and makes it available to the installation. The installation specifies the name of the ESMT to IMS by including it on the definition of the external subsystem to IMS.

Resource translation table (RTT)

IMS uses a PSB (program specification block) to define the DL/I resources required by an application program. For an MPP, the PSB name is the same as the application program name; for a BMP or IFP, the PSB name can be different. The external subsystem can use a name other than the PSB name or the IMS application program name for the entity it uses to define the external subsystem resources required by the application program. If the external subsystem uses a different name, the external subsystem can provide a resource translation table (RTT) to map either PSB names or application program names to its entity names.

The external subsystem creates the RTT and makes it available to the installation. The installation specifies the name of the RTT on the definition of the external subsystem to IMS. IMS loads the RTT when it loads the ESAP.

The external subsystem is responsible for doing the actual mapping. IMS does not access the RTT; it merely loads the table and makes its address available to the ESAP. IMS does not prescribe the format of the RTT.

Related concepts

“Creating the external subsystem module table” on page 121
The external subsystem creates the external subsystem module table (ESMT) to supply definitions of the external subsystem modules that IMS is to load.

How external subsystems are specified to IMS

In an IMS.PROCLIB member, define all external subsystems that are to be accessed by IMS applications. The EXEC statement of the control region points to this member with the SSM parameter.

For each external subsystem defined to IMS, specify in the IMS.PROCLIB member:

- The external subsystem type
- The z/OS name of the external subsystem
- The name of the external subsystem module table (ESMT) that specifies the modules in the external subsystem attachment package (ESAP)

- The language interface token (LIT) that IMS uses to route application calls to the external subsystem
- The name of a resource translation table (RTT) supplied by the external subsystem, if needed, to identify the external resources required by IMS application programs
- The command recognition character (CRC) that IMS uses to route operator commands to the external subsystem
- The region error option (REO) code indicating the action to be taken when application calls to the external subsystem cannot be processed. When a Resource Translation Table (RTT) is used, the OPTION value specified in the RTT overrides the REO option in the SSM member.

You must also supply the external subsystem-supplied tables (ESMT and RTT) in the appropriate load module library.

You have the option to supply external subsystem definitions for dependent regions. If the SSM EXEC parameter is not specified for these types of dependent regions, the region can access all subsystems defined to the control region. If the SSM EXEC parameter is specified, the dependent region can access only those subsystems defined in the identified PROCLIB member. (The subsystems also must have been defined to the control region.) Use a dummy PROCLIB member (one having no definitions) if the dependent region is not to have access to any external subsystem.

Note: JMP and JBP regions can also access control-region-defined Db2 for z/OS subsystems that use the COORD=RRS parameter in the IMS.PROCLIB member. In the DFSJMP and DFSJBP procedures, add a DD statement (DFSDB2AF) that points to the Db2 for z/OS libraries.

The following tasks must be performed to attach an external subsystem to IMS.

To attach an external subsystem to IMS:

1. Define external subsystems to IMS:
   a) In the IMS procedure library (IMS.PROCLIB), add a member that contains the information about each external subsystem with which IMS communicates.
   b) On the EXEC statement of the IMS control region or the dependent region, specify in the SSM parameter the PROCLIB member that you created in the previous step.

      If you are using a Db2 for z/OS group name to access Db2 for z/OS databases, you must specify the group name in the EXEC statement of the dependent region. A Db2 for z/OS group name cannot be specified in the EXEC statement for the IMS control region.
2. Define a language interface module if you want to use one other than the IMS-supplied one.
3. In the IMS OPTIONS statement, specify whether you want tracing of the external subsystem link.
4. For the external subsystem, provide the ESMT and optionally the RRT.
5. Ensure that the external subsystem modules and databases used by IMS are in appropriate APF-authorized libraries.

Related concepts

Accessing external subsystem data (System Definition)

The basics of attach processing

An external subsystem is attached to an IMS subsystem by means of a connection established from the IMS control region to the external subsystem.

A connection is also established from each dependent region that accesses the external subsystem. IMS is responsible for initiating these connections.

Subsystem connections

The connection between an IMS application program and the external subsystem is called a *thread*. Application threads are two-way communication paths between IMS application programs and external subsystem resources.

An application program can have more than one thread since it can access more than one external subsystem in one execution. However, access to multiple subsystems of the same type (multiple instances of the same subsystem type) while supported by IMS, might not be supported by the external subsystem product.

Establishing connections

IMS uses an 'identify' process to establish a connection to the external subsystem.

IMS activates an Identify exit routine contained in the ESAP to identify the control region or dependent region TCB to the external subsystem. The external subsystem can then monitor IMS TCBs in order to respond to IMS abnormal terminations. A connection is established upon successful completion of the identify process, in other words, once the region has been successfully identified to the external subsystem.

IMS provides a notify message mechanism so that if the external subsystem has not been started when IMS attempts to connect the control region, the external subsystem, once started, can notify IMS to establish the connection. If the external subsystem makes use of the notify capability, the order in which IMS and the external subsystem are started is not important.

The connection from the control region is established first before any dependent region connections are established. If the control region connection has not been established when a dependent region is started, the dependent region does not identify itself to the external subsystem. IMS uses a hierarchical relationship between control region and dependent region connections to allow the control region to act as recovery coordinator for dependent regions. If a dependent region fails, the control region takes recovery actions on its behalf.

The external subsystem can optionally provide an Initialization exit routine. IMS activates the Initialization exit routine, if provided, during control region and dependent region initialization before the region identifies itself to the external subsystem. This exit routine allows the external subsystem the chance to do any initialization processing it requires before each connection being established.

IMS can establish the control region connection automatically during control region initialization, provided the external subsystem has been started. However, the connection can be delayed to a later time. If an Initialization exit routine is not provided, or if the exit routine returns the appropriate return code, the control region identify is not done automatically. In this case, the external subsystem can activate the Subsystem Startup Service provided by IMS when it wants the connection established. Or, IMS attempts to establish the connection when a dependent region is ready to identify itself.

IMS also establishes the control region connection in response to a **/START SUBSYS** operator command.

User authorization processing

After a dependent region connection has been established, a signon process is performed to inform the external subsystem of the user ID associated with the IMS transaction being processed by the region. IMS activates a Signon exit routine provided by the external subsystem for this purpose. This initial signon for the region must be successful in order for a thread to be created for the application.

Signon processing can occur again during application program execution (that is, after the thread has been created). The Signon exit routine is activated for each message processed by the application program. The initial signon performed after the dependent region identify is related to the first message processed by the application (first get unique call to the message queue). Each subsequent message processed causes the Signon exit routine to be activated again to pass the new user ID. In the case of multiple mode transactions, this means that multiple signons can occur without intervening commit processing.

The external subsystem supplies a Signoff exit routine which IMS activates before terminating the dependent region connection. In the case of multiple signons for an application, signoff processing does not precede a new signon for a new message. A new signon rather replaces the previous.

Application threads

When the application program issues its first call for data resources owned by an external subsystem, a thread is created to connect the application to the external subsystem.

IMS activates a Create Thread exit routine supplied in the ESAP to identify the application program to the external subsystem. The external subsystem is expected to prepare to receive data requests from the specific application program as necessary (that is, reserve resources, create a processing structure, and so on). When the application terminates, IMS activates a Terminate Thread exit routine to terminate the thread.

Terminating connections

A Terminate Identify exit routine must be provided in the ESAP. IMS activates this exit routine when a connection is to be terminated.

Termination of the control region connection can be initiated by IMS, by the external subsystem, or by operator command (**/STOP SUBSYS**). IMS terminates the connection when it is shutting down. The **/STOP SUBSYS** command causes the connection to be terminated and also puts it in stopped status. IMS does not allow the connection to be reestablished until a **/START SUBSYS** command has been processed.

The external subsystem can request that the control region connection be terminated in one of two ways. One way is by posting a termination ECB. IMS provides, on the Identify exit routine invocation, the address of an ECB that is expected to be used by the external subsystem when it is terminating. When the external subsystem posts the termination ECB, IMS, after allowing dependent region connections to quiesce, terminates the connection and also puts the connection in stopped status (as it does for the **/STOP SUBSYS** command). The second way that the control region connection can be terminated is by activating the IMS-supplied Subsystem Termination Service from an external subsystem exit routine.

After activating the Terminate Identify exit routine in the control region, IMS activates the Subsystem Termination exit routine supplied in the ESAP. This exit routine, which can be thought of as the reverse of the Initialization exit routine, might be used by the external subsystem to reset work areas or free storage, for example.

A dependent region connection is maintained for as long as the region is active unless IMS has been requested, either by the external subsystem or by an IMS **/STOP SUBSYS** command, to terminate the (control region) connection. In general, it is only when IMS has been requested to terminate the connection that the external subsystem Terminate Identify exit routine is driven for dependent regions. Thus, the exit routine is not necessarily activated when a dependent region terminates. This is true also for the Signoff exit routine. Terminate Identify exit routine invocation always follows Signoff exit routine invocation.

The external subsystem is expected to monitor, through the z/OS end-of-task exit routines, the IMS TCBs identified to it and to perform the necessary signoff and terminate identify processing when an identified TCB ends.

Terminate Thread exit routine invocation always precedes normal termination of the dependent region connection if the region had a thread to the external subsystem. Thus the Terminate Thread exit routine is activated before the Signoff and Terminate Identify exit routines are activated (if they are) or before the dependent region is terminated.

Since IMS does not allow dependent region connections to exist unless the control region has a connection, the Terminate Identify exit routine is not activated for the control region until after each dependent region has either terminated or had its Terminate Identify exit routine activated.

The Subsystem Termination exit routine is not activated for dependent regions.

Inquiry parameter processing

The INQ parameter is only checked when the IMS transaction issues a Create Thread exit routine.

The INQ parameter on subsequent transactions is not checked. Therefore, if any updates are to be done in a Fast Path region between the Create Thread exit routine and the Terminate Thread exit routine, the inquiry flag in the first transaction must be INQ=NO.

For example, if the first transaction that calls Db2 for z/OS from a given Fast Path region is only going to read the Db2 for z/OS data and not update it, the transaction will set the INQ=YES flag in the Create Thread Parameter list indicating that this first transaction and all subsequent transactions in that Fast Path region are treated as inquiry only transactions. If a subsequent transaction running under the same Fast Path region calls Db2 for z/OS for update, the thread to that Fast Path region will still be set to INQ=YES, even though the transaction is correctly defined as INQ=NO. This will result in an SQLCODE817 error.

Application call processing

After a thread from the application program to the external subsystem has been created, application calls for external data resources are passed to the Normal Call exit routine supplied in the ESAP.

The language interface bound with the application provides the language interface token (LIT) for the external subsystem when it activates IMS to process calls to the external subsystem. The installation specifies a unique LIT for each external subsystem it defines to IMS. IMS matches the LIT provided by the language interface stub with the LIT specified in the definition to route the call to the external subsystem.

Resource coordination

IMS, as recovery coordinator, directs commit processing for updates to external subsystem resources initiated by IMS application programs. IMS uses a two-phase commit process to synchronize resources across external subsystems. External subsystems are participants in the process.

In the first phase of the commit process for an application, IMS polls the participants for a vote as to whether or not they are prepared to commit the updates. In the second phase, IMS directs the participants to commit or to abort. If all participants voted 'yes' on the first phase, IMS directs them to commit on the second phase; otherwise, IMS directs them to abort.

When an external subsystem determines that its resources are associated with non-update transactions (for which commit processing is not necessary), the external subsystem can perform all commit processing during the first phase, eliminating the need for the second phase. In this case, the external subsystem returns to IMS from the Commit Prepare exit routine with return code X'C' indicating that the first phase successfully completed and the second phase is not required. IMS will not initiate the second phase of commit processing for this external subsystem.

IMS uses a 16-byte recovery token to identify a unit of work across one or more subsystems. The recovery token for a unit of work is initially passed on the Signon exit routine invocation.

When application updates are to be committed, IMS activates the Commit Prepare exit routine supplied by the external subsystem. The associated recovery token is passed on the invocation. The external subsystem indicates, by the return code from the exit routine, whether or not it is prepared to perform commit processing for the recovery token. When an application is executing in a Distributed Syncopate environment (also known as a Protected Conversation environment) and requires a subsystem SIGNON, IMS obtains the XID token and places its address in the exit parameter list before calling the subsystem's SIGNON exit.

For the second phase of the commit process, if it is required, IMS can activate either of three external subsystem exit routines: the Commit Continue exit routine, the Abort Continue exit routine, or the Terminate Thread exit routine. When the application is not terminating and all participants are prepared to commit, IMS drives the Commit Continue exit routine. At completion of the commit process, the application will continue processing the current PSB on the existing thread. When the updates are to be aborted but the application is not terminating, or being terminated, the Abort Continue exit routine is activated. In this case, the application will continue processing under the same recovery token.

The external subsystem Terminate Thread exit routine must be able to process the second phase of commit. At application termination, IMS passes the recovery token and a commit option on the Terminate Thread exit routine invocation. The commit option indicates whether to commit or abort outstanding updates.

When IMS, the external subsystem, or an application program terminates abnormally, units of work that have not been committed or aborted are left outstanding. To resolve outstanding units of work, IMS activates the external subsystem Resolve Indoubt exit routine. IMS always activates the Resolve Indoubt exit routine at least once after establishing the control region connection. IMS activates the exit routine once for each outstanding recovery token indicating whether to commit or abort the unit of work. When there are no units of work to be resolved or when IMS has exhausted the list of outstanding recovery tokens, IMS activates the exit routine to inform the external subsystem of that fact. When IMS encounters an outstanding recovery token associated with z/OS Resource Recovery Services, IMS will delay the subsystem Resolve Indoubt exit call until RRS or the IMS user has indicated (ABORT or COMMIT) which action to take. When called for RRS Resolve Indoubt, it is the subsystem's responsibility to ensure that recovery tokens are resolved in their proper order.

IMS maintains outstanding recovery tokens across normal (warm) and emergency restarts of IMS, and reconnections of the subsystems. IMS permits a connection without all recovery tokens being resolved (that is, the Resolve Indoubt exit routine return code can indicate that the recovery action was not taken). IMS destroys outstanding recovery tokens when it is cold started.

The Resolve Indoubt exit routine is also used to coordinate resources in the event of abnormal termination of an application program. Following an application program abend, the exit routine is activated from the control region if the application had a thread connection to the external subsystem.

You can also configure a Fast Database Recovery (FDBR) region to recover work on the external subsystem. When a FDBR region is monitoring an IMS system that fails, it receives information about indoubt work on the external subsystem from the ESAF Indoubt Notification exit routine (DFSFIDN0). Units of work associated with z/OS Resource Recovery Services are not recovered by FDBR.

Related concepts

Fast Database Recovery (FDBR) regions (Operations and Automation)

External subsystem command support

IMS provides a command, /SSR, which allows the IMS operator to send commands to the external subsystem.

To receive commands from IMS the external subsystem must supply a Command exit routine. IMS passes the command contained in the /SSR input to this exit routine. AOI (automated operator interface) programs can also send commands to external subsystems using /SSR. The /SSR command input contains identification, a command recognition character (CRC), of the external subsystem to which the command is directed. The CRC for a subsystem is specified as part of the definition of the subsystem to IMS.

Related tasks

“How external subsystems are specified to IMS” on page 114
In an IMS.PROCLIB member, define all external subsystems that are to be accessed by IMS applications. The EXEC statement of the control region points to this member with the SSM parameter.

Related reference

/SSR command (Commands)

IMS services available to the ESAP

IMS provides exit routines that an external subsystem can activate to access certain IMS system services.

The external subsystem can:

- Request that a connection be initiated (Subsystem Startup Service).
- Request that connections be quiesced (Subsystem Termination Service).

- Have a log record written to the IMS log (Log Service).
- Have a message sent to an IMS destination (Message Service).

Related concepts
Exit routines (Exit Routines)

Chapter 8. Creating the external subsystem module table

The external subsystem creates the external subsystem module table (ESMT) to supply definitions of the external subsystem modules that IMS is to load.

External subsystem exit routine modules, as well as any other modules needed in the ESAP, are defined in the ESMT. The installation provides the name of the ESMT to IMS as part of the definition of the external subsystem. During initialization for attach processing in the control and dependent regions, IMS loads the ESMT and then loads the modules defined therein.

The external subsystem optionally provides in the ESMT definitions of work areas needed for its ESAP. If work area definitions are provided, IMS obtains the specified work area storage in each region after loading the defined modules.

IMS provides two macros to be used by the external subsystem to create the ESMT. The DFSEMODL macro is used to define the ESAP modules that IMS is to load. The DFSEWAL macro is used to define the work areas that IMS is to create. A series of DFSEMODL statements defining modules, optionally followed by DFSEWAL statements defining work areas, and ending with a DFSEMODL statement specifying END=LAST, generates the table.

DFSEMODL macro

The external subsystem module table (ESMT) is generated from a series of DFSEMODL statements, one for each module definition.

In addition to module definition information, information about the control block that is to contain the addresses of the modules when they are loaded is also supplied on DFSEMODL statements. IMS creates this control block before it loads the modules.

IMS provides the capability for up to three sets of modules to be loaded and anchored on separate control blocks. Accordingly, the ESMT consists of one to three subtables, each containing the specifications for a set of modules and their module address control block. The module address control block for external subsystem exit routines is the EEVT (external entry vector table).

When the module address control block is created, IMS stores its address into a source control block, which is the EEVTP (EEVT prefix).

The format of the DFSEMODL macro is:

```
(label)   DFSEMODL DSNAME=,SOURCE=,MODNAME=,DSLABEL=,
                   SUBPOOL=,OPTION=,END=
```

where:

(label)
Is optional. If coded, the macro generates ESMT subtable labels. The last label on a macro statement in the series from which a subtable is generated is used as the subtable label.

The following parameters provide control block information and need only be specified once per subtable (for example, on the first DFSEMODL statement in the series). If specified on more than one statement, the first specifications encountered are used in generating the table.

DSNAME=
(p1,p2,p3)

p1
Name of the module address control block. The name must be specified (at least on one DFSEMODL statement) for each ESMT subtable.

© Copyright IBM Corp. 1974, 2020

p2
Module address control block size. The size must be specified. IMS obtains storage of the specified size to create the module address control block.

p3
Subpool number for the module address control block storage request. This parameter is optional. If 251 is not specified, IMS obtains the storage from subpool 230.

SOURCE=
(p1,p2)

p1
Name of the source control block. This parameter is required. DFSEEVTP must be specified. (See the following discussion.)

p2
Label in the source control block of the location to store module address control block address. This parameter is required.

The following parameters provide module definition information.

MODNAME=
Name of the module IMS is to load. MODNAME must be specified on all DFSEMODL statements that do not specify the END parameter. (END can be specified with or without MODNAME.)

DSLABEL=
Label in the module address control block of the location to store the module address after it is loaded. DSLABEL must be specified (when MODNAME is specified).

SUBPOOL=

For resources that reside on a PDS data set:
The subpool into which IMS is to load the module. SUBPOOL must be specified when MODNAME is specified. For the control region, IMS loads the module into the subpool specified. For dependent regions, IMS loads the module into subpool 251 if SUBPOOL=251 is specified. Otherwise, the module is loaded into subpool 230. Valid specifications are 0, 229, 230, 231, 241, 251, 252.

For modules that reside on a PDSE data set:
The SUBPOOL parameter is not used. Modules residing on a PDSE are loaded in one of the following methods:

- Modules that are linked as reentrant (RENT) are loaded into subpool 252, key 0. These modules are not fetch-protected.
- Modules that are linked as not reentrant (NORENT) are loaded into subpool 251, TCBKEY, and are fetch-protected. You must ensure that the correct protect key is in use before accessing these modules.

OPTION=
(p,p)

This parameter is optional. Two options, NOCTL and NODEP, are supported. (Position of an option in the subparameter list is not important.)

NOCTL
The module is not to be loaded in the control region.

NODEP
The module is not to be loaded in dependent regions.

The END parameter controls ESMT generation.

END=

YES
Must be specified to indicate the end of a subtable in the ESMT being generated. END=YES is used only when the ESMT is to contain more than one subtable. It is specified to end each subtable except the last (or only). DFSEMODL statements for the next subtable in the ESMT are to follow the END=YES specification.

LAST
Must be specified on the last DFSEMODL definition statement for the ESMT being generated. The next DFSEMODL or DFSEWAL statement (if any) causes a new ESMT generation to be started.

You must bind the ESMT module into a program library (SDFSRESL) using a binder ENTRY statement that specifies MAINEP as the entry point. A table definition header is generated at the end of the ESMT module. The ENTRY statement allows IMS to correctly reference the header for subsequent processing.

DFSEMODL supports an execute form (MF=E) for internal use only. It cannot be used for ESMT generation. The list form (MF=L) described is the default.

Mapping DSECTs for the module address and source control blocks must be included in the ESMT generation source, otherwise the assembly will fail.

The following restrictions apply to external subsystems:

- Specifying the source control block.

  DFSEEVTP must be specified as the source control block name (SOURCE(p1)) for all IMS-defined subtables. Otherwise, although DFSEMODL accepts other specifications, the module load process will fail, prohibiting a connection for the region experiencing the failure. The user must use the EEVTP mapping layout, as this is the layout that IMS expects.
- Defining subsystem exit routine modules.

  DFSEEVT must be specified as the module address control block name (DSNAME(p1)) for the subtable that contains the subsystem exit routine module definitions.

  The module address control block size (DSNAME(p2)) must be specified according to the size indicated by the EEVT mapping (EEVTLGTH) as shown in "Control block mapping" in *IMS Version 15 Exit Routines*.

  EEVPEEA must be specified as the label in the source control block (SOURCE(p2)) to anchor the module address control block (EEVT). IMS does not check for this (nor does the macro) but uses the offset generated from the label specified to store the address. If the offset is incorrect, IMS will not be able to activate the exit routines.

  The label (DSLABEL) specified for a particular subsystem exit routine module is used to generate the offset that IMS uses to store the exit routine address in the module address control block (for example, in the EEVT). Thus these labels must be specified according to the EEVT mapping.
- Generating additional subtables.

  The ESMT must always have one subtable containing definitions for exit routine modules. The external subsystem could choose to have other modules needed in its ESAP anchored on a separate control block, which means that another subtable would be generated.

  Although the DFSEMODL macro does not restrict the number of subtables that can be generated, problems can occur during processing if more than three (3) are generated. For each subtable, IMS creates a module address control block and stores its address in the EEVTP. There are only three fields in the EEVTP that could be used as anchors for these control blocks, one of them being the anchor for the EEVT.

  The EEVTLDIR and EEVPEWA fields are not used by IMS and thus are available for this purpose. The discussion on defining external subsystem work areas in “DFSEWAL macro” on page 124 suggests how EEVPEWA might be used to anchor a work area address control block.
- Defining external subsystem-unique modules.

  If the ESAP needs non-IMS exit routine modules (for example, modules that the external subsystem activates without any knowledge of or support from IMS), the external subsystem can define these modules in an additional subtable as previously discussed. The external subsystem must supply the mapping DSECT for the module address control block for these modules.

  Recommendation: Do not define other modules in the subtable containing exit routine modules or extend the size of the EEVT to include their addresses. The EEVT is an IMS control block which IMS

reserves the right to extend at any time, which could require the external subsystem to regenerate the ESMT and re-compile modules.

DFSEWAL macro

The work areas that IMS is to create for the external subsystem must be defined by including DFSEWAL macro statements along with the DFSEMODL statements provided for ESMT generation.

The DFSEWAL statements, one for each work area defined, follow the DFSEMODL statements, except that the last statement in the series must be a DFSEMODL statement specifying END=LAST. The DFSEWAL statements cause a table of work area definitions to be built in the generated ESMT.

Work areas can be defined in each subtable generated in the ESMT. At least one module must be defined in each subtable. If a subtable is generated containing only work area definitions, an error occurs during IMS processing of the ESMT.

IMS creates the work areas defined in a subtable after loading the modules defined in the subtable. IMS stores the addresses of the created work areas in a work area list control block. This control block is also defined by the DFSEWAL macro and can either be contained in the module address control block for the subtable or be created as a separate control block. For this discussion, EWAL is used to refer to the external subsystem work area list control block.

Recommendation: Contain the EWAL in the module address control block rather than creating it as a separate control block. When IMS creates the EWAL, its address is not (explicitly) provided to the external subsystem. If, instead, the EWAL is contained in the module address control block, which IMS anchors in the EEVTP, the external subsystem specifies its location (with DFSEWAL) and thus knows how to access it. (When IMS creates the EWAL, it stores its address in the in-storage ESMT for internal use. The format of the ESMT is not included in the documented attach interface.) The following figure shows a representation of the relationship between the EWAL, EEVTP, and EEVT.

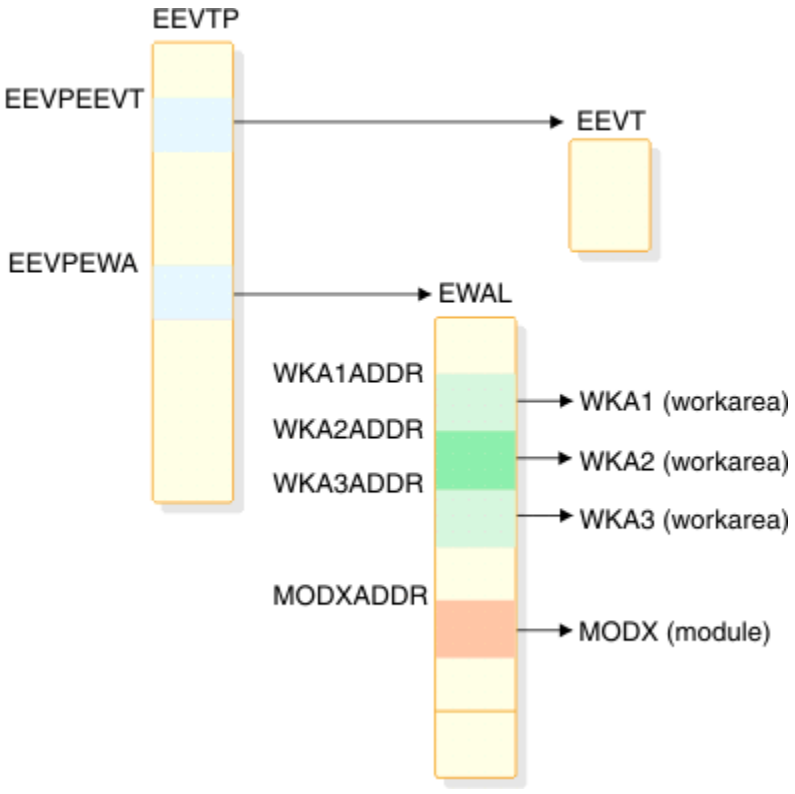


Figure 16. EWAL, EEVTP, and EEVT relationship

If the external subsystem wants IMS to create work areas for its ESAP, it should define two (possibly three) subtables in the ESMT. Modules definitions would be contained in one subtable. The module address control block for this subtable is the EEVT. The second subtable would contain work area

definitions. The module address control block for this subtable would either contain the EWAL or be used as the EWAL, and would be anchored in the EEVTP along with the EEVT. Modules could be defined in two subtables: one for exit routines and one for other external subsystem modules that are activated by exit routines.

The second larger block of example code below illustrates how the external subsystem might specify work area definitions.

The format of the DFSEWAL macro is as follows:

```
DFSEWAL DSNAME,SOURCE=,WALSP=,NAME=,DSLABEL=,
        SUBPOOL=,LV=,OPTION=
```

where:

The following parameters provide control block information and need only be specified once per subtable (for example, on the first DFSEWAL statement in the series). If specified on more than one statement, the first specifications encountered are used in generating the table.

DSNAME=
(p1,p2)

p1
Name of the work area list control block mapping DSECT. The DSECT name must be specified. If IMS creates the work area list, this name is given to the job pack entry for the storage acquired.

p2
Work area list size. If the size is specified, IMS obtains storage of the specified size to create the work area list. If the size is not specified IMS does not create the work areas unless the source control block for the work area list (DFSEWAL SOURCE(p1) specification) is the module address control block specified for the modules defined in the subtable (DFSEMODL DSNAME(p1)). (See the following discussion.)

SOURCE=
(p1,p2)

p1
DSECT name for the control block in which the work area list is to be anchored. This parameter must be specified. (See the following discussion).

p2
Label in the source control block DSECT of the location that is to contain the work area list. This parameter is required. IMS does not store the work list area address into this control block. (See the following discussion.)

WALSP=
Subpool number for the work area list storage request. This parameter is optional. If WALSP=251 is not specified, IMS obtains the storage from subpool 230.

The following parameters provide work area definition information and must be specified on each DFSEWAL statement.

NAME=
The name given to the job pack directory entry created for the work area storage acquired. This parameter is required.

DSLABEL=
Label in the work area list control block DSECT of the location into which IMS is to store the work area address. DSLABEL must be specified.

SUBPOOL=
Subpool from which IMS is to obtain storage for the work area. The subpool must be specified. IMS acquires subpool 251 storage if SUBP00L=251 is specified; otherwise, the work area is created in subpool 230. The macro allows 0, 229, 230, 231, 241, 251, or 252 to be specified.

LV=
Work area size. The size must be specified.

OPTION=
(p,p)

This parameter is optional. Two options, NOCTL and NODEP, are supported. (Position of an option in the subparameter list is not important.)

NOCTL
Work area is not to be created in the control region.

NODEP
Work area is not to be created in dependent regions.

Mapping DSECTs for all referenced control blocks must be included in the ESMT generation source, otherwise the assembly will fail.

The source control block DSECT name and label must be specified. However, IMS does not store the EWAL address into this control block.

To indicate that the EWAL is to be contained in the module address control block:

- The size for the EWAL (DFSEWAL DSNAME(p2)) must not be specified.
- The module address control block DSECT name (DFSEMODL DSNAME(p1)) must be specified as the EWAL source control block DSECT name (DFSEWAL SOURCE(p1)).
- The source control block label (SOURCE(p2)) must specify the location of the work area list in the module address control block.

If the size for the EWAL is specified, IMS obtains storage for the EWAL without checking if the module address control block was specified as the EWAL source. If the EWAL size is not specified and the module address and EWAL source control block DSECT names do not match, IMS does not create the work areas. (IMS does not know the address of the source control block. IMS does not indicate that the work areas were not created.)

IMS reserves the EEVPEWA field in the EEVTP control block for the address of an EWAL. The following code illustrates how definitions can be specified by an external subsystem to anchor a work area list in this field. What really happens is that a module address control block is created, anchored at EEVPEWA, and used as the EWAL.

```
      DFSEMODL DSNAME=(DFSEEVT,68,230),SOURCE=(DFSEEVTP,EEVPEEA),
               MODNAME=INITEXIT,DSLABEL=EEVTINIT,SUBPOOL=230
         DFSEMODL MODNAME=IDEXIT,DSLABEL=EEVTID,SUBPOOL=230
         DFSEMODL MODNAME=RIDEXIT,DSLABEL=EEVTRID,SUBPOOL=230,
                  OPTION=NODEP
                       .       .       .
                       .       .       .
                       .       .       .
         DFSEMODL MODNAME=CMDEXIT,DSLABEL=EEVTCMD,SUBPOOL=230,
                  OPTION=NODEP
         DFSEMODL END=YES
         DFSEMODL DSNAME=(ESSEWAL,40,230),SOURCE=(DFSEEVTP,EEVPEWA),
                  MODNAME=MODX,LABEL=MODXADDR,SUBPOOL=230
         DFSEWAL  DSNAME=(ESSEWAL),SOURCE=(ESSEWAL,ESSEWAL),WALSP=230,
                  NAME=WKA1,DSLABEL=WKA1ADDR,SUBPOOL=230,LV=200
         DFSEWAL  NAME=WKA2,DSLABEL=WKA2ADDR,SUBPOOL=230,LV=100
         DFSEWAL  NAME=WKA3,DSLABEL=WKA3ADDR,SUBPOOL=230,LV=100
         DFSEMODL END=LAST
```

Notes to example:

- The existence of a DSECT named ESSEWAL created by the external subsystem to map the EWAL is assumed.
- Two subtables are defined for completeness:
  - The first subtable contains exit routine module definitions.
  - The second subtable contains work area definitions:
    - A module is defined in this subtable with EEVPEWA specified as the anchor field for the module address control block. (If the external subsystem does not really want a module loaded for this subtable, both the NOCTL and NODEP options can be specified.)

- The module address control block DSECT, ESSEWAL, is specified as the EWAL source control block DSECT and the EWAL size is not specified, indicating that the EWAL is to be contained in the module address control block.
- ESSEWAL is also specified as the label in the source block for the EWAL, indicating that the EWAL starts at offset zero in the module address control block. Thus, the module address control block itself is the EWAL, anchored at EEVPEWA in the EEVTP.

Chapter 9. IMS External Subsystem Attach Facility processing

The IMS External Subsystem Attach Facility performs processing during control region initialization.

Loading the External Subsystem Attachment Package

During control region initialization, IMS loads external subsystem-supplied tables using the table names specified by the installation (in the external subsystem definition member in IMS.PROCLIB).

IMS loads the external subsystem module table (ESMT) and then loads the external subsystem modules defined in the table. The resource translation table (RTT) is also loaded, if provided. If an error occurs during this process, IMS puts the subsystem in 'stopped' status and does not establish a connection. However, IMS will reaccess the definition (PROCLIB) and reattempt this process if a **/START SUBSYS** command is received.

Possible errors are:

- Unable to process the PROCLIB member
- Unable to open the external subsystem load library
- Unable to load the ESMT (incorrect name specified, not in library)

IMS stores the addresses of the ESMT and the RTT in the EEVTP control block fields EEVPESMT and EEVPRTTA, respectively.

Creating the EEVT control block

IMS creates and initializes the EEVT control block based on information contained in the external subsystem module table (ESMT) that is generated from DFSEMODL macro statements.

The size and subpool for the EEVT storage request are obtained from the ESMT. The EEVT is created in subpool 230 unless subpool 251 is specified in the ESMT. The external subsystem must ensure that the size specified for the EEVT is at least as large as the size indicated in the IMS EEVT mapping.

IMS stores the EEVT address into the EEVTP control block based on an offset specified in the ESMT. The external subsystem must ensure, therefore, that the offset generated in the ESMT (using the DFSEMODL macro) points to the EEVPEEA field in the IMS EEVTP mapping.

IMS does not check whether the EEVT pointer field, EEVPEEA, in the EEVTP is initialized by this process. In fact, the offset in the ESMT could cause IMS to store the address into some other field in the EEVTP designated for some other use, possibly causing a problem. Thus, **the external subsystem must ensure that the correct offset is generated into the ESMT**.

This process allows the external subsystem to specify another set of modules for IMS to load (IMS would not activate these modules). Both lists of module addresses, one being the EEVT, would be anchored in the EEVTP.

IMS does not use the EEVTLDIR field. Actually more than two sets of modules could be defined in the ESMT (subtables) and loaded by IMS except that there are not enough fields in the EEVTP to anchor the address lists.

Related concepts

“Creating the external subsystem module table” on page 121

© Copyright IBM Corp. 1974, 2020

The external subsystem creates the external subsystem module table (ESMT) to supply definitions of the external subsystem modules that IMS is to load.

Loading external subsystem modules

As IMS loads external modules defined in the ESMT, the module addresses are stored in the EEVT.

The module definitions provide the offsets to the locations in the EEVT for the addresses. IMS does not check whether or not required exit routine addresses have been set by the module loading process. If the external subsystem chooses, the ESAP can set exit routine addresses in the EEVT once IMS has passed control to it. For example, the external subsystem can provide multiple exit routines in one load module and have the ESAP set the individual exit routine module addresses.

Whether through module definitions in the ESMT or through ESAP processing, the external subsystem must ensure that the address of an exit routine is present in the EEVT when IMS needs to activate the exit routine. Exit routine addresses must be placed in the EEVT according to the IMS EEVT mapping.

Some of the exit routines prescribed by IMS are activated only in the control region; some are activated only in the dependent regions. The external subsystem can indicate, in the module definition, if a module is not to be loaded in the control region or in dependent regions. Exit routine module definitions should specify loading according to the following:

- Exit routines activated in the control and dependent regions:
  - Identify
  - Initialization
  - Terminate Identify
- Exit routines activated only in the control region:
  - Command
  - Echo
  - Resolve Indoubt
  - Subsystem Termination
- Exit routines activated only in dependent regions:
  - Abort Continue
  - Associate Thread
  - Commit Continue
  - Commit Prepare
  - Commit Verify
  - Create Thread
  - Normal Call
  - Signoff
  - Signon
  - Subsystem Not Operational
  - Terminate Thread

In the control region, IMS loads external subsystem modules into the subpools specified in the module definitions. If subpool 251 is specified for a module in dependent regions, IMS loads the module in subpool 251; otherwise, it is loaded in subpool 230.

An external subsystem uses only those exit routines that it needs to communicate with IMS, although some exit routines are required and others are optional. When a required exit routine does not exist, IMS generates an error message when it tries to call the exit routine and terminates the connection with the external subsystem.

If your external subsystem does not need the function that an exit routine is designed to perform, you can write the exit routine so that one exists when IMS calls it but that so no operations are performed. (An exit routine can contain common code, such as SR 15,15 and BR 14 logic, which ESS branches to when the exit routine is called and which does not perform any specific operation.) During processing of the Initialization exit routine, the external subsystem can update the addresses in the DFSEEVT DSECT (from both the control region and dependent regions, if necessary) and point to these exit routines. This action allows IMS to function normally yet not issue error messages and terminate an external connection if an exit routine does not exist.

Creating work areas for the ESAP

After loading the ESAP modules, IMS obtains work area storage for the ESAP if work area definitions are contained in the ESMT.

The IMS DFSEWAL macro is used to generate work area definitions in the ESMT.

The process IMS uses to create the work areas is similar to the process used to load ESAP modules except that IMS can either:

- Create the control block for the work area addresses (as it creates the EEVT for the ESAP module addresses), or
- Store the work area addresses into the same control block that has the module addresses.

The intended use of the EEVPEWA field in the EEVTP is to hold the address of a control block referred to as the external subsystem work area list (EWAL) that contains the addresses of the work areas created for the ESAP. However, the external subsystem must have provided the appropriate specifications in the ESMT to cause IMS to store the address of the EWAL in this field.

IMS creates each work area either in the control region or in dependent regions or both, according to the definition. Storage is obtained in subpool 251, if specified; otherwise it is obtained in subpool 230.

Related concepts

“Creating the external subsystem module table” on page 121
The external subsystem creates the external subsystem module table (ESMT) to supply definitions of the external subsystem modules that IMS is to load.

Initiating the external subsystem connection

IMS automatically connects to the external subsystem during control region initialization processing (for example, when IMS is started) unless the external subsystem chooses to defer the control region identify to a later time.

The external subsystem defers the connection by returning from the control region Initialization exit routine with return code 4 (do not identify), or by not providing an Initialization exit routine for the control region.

If the external subsystem uses the notify message mechanism provided by IMS (and if the external subsystem is not up when IMS activates the Identify exit routine) the connection is automatically established when the external subsystem is started. Return Code 4 from the Identify exit routine causes IMS to wait for the external subsystem to send the notify message passed to the exit routine, and to reactivate the exit routine when the message is received.

If the notify mechanism is not used, the Identify exit routine should return with return code 12, in which case the connection is put in stopped status. Stopped status must be removed by a **/START SUBSYS** command before IMS will establish a connection.

Once the connection has been established, IMS performs Resolve Indoubt processing to resolve any outstanding recovery tokens with the external subsystem. If outstanding recovery tokens exist and a Resolve Indoubt exit routine was not supplied, IMS terminates and stops the connection; otherwise dependent regions are allowed to connect to the external subsystem.

Deferring the control region identify

The external subsystem can defer the control region identify if it prefers to have the connection established at some later time.

IMS schedules and gives control to application programs whether or not a connection exists to the external subsystem. The external subsystem thus could choose to wait until an application program call has to be serviced (first call for its resources) before connecting to IMS. Of course, to process calls, the control and dependent region connections and the application thread must exist.

When the control region identify is deferred, the identify is done when:

- The external subsystem, with an exit routine, activates the IMS Subsystem Startup Service.
- An MPP or IFP dependent region that can access the external subsystem is started.
- A **/START SUBSYS** command naming the external subsystem is processed.

Using the IMS Subsystem Startup Service

When the control region identify is deferred, the external subsystem can activate the IMS Subsystem Startup Service when it wants the control region Identify exit routine to be driven.

To be more specific, if a connection does not exist when the first application call for external subsystem services is processed by a dependent region, IMS does not automatically attempt to identify. The external subsystem must activate the Startup Service to establish the connection (if it **wants** to process application calls).

The Startup Service is also used to establish dependent region connections. When an external subsystem call from an application is processed before the control region or dependent region has been identified to the external subsystem, the dependent region activates the Subsystem Not Operational exit routine. The external subsystem is expected to call the Subsystem Startup Service from this exit routine to establish the connection.

When activated, the Startup Service establishes the control and dependent region connections, if the control region identify has not been done. If the control region identify has been done, it establishes only the dependent region connection. If IMS is waiting for the external subsystem to send the notify message, which it accepted on a previous Identify exit routine invocation, the Startup Service returns an error return code and does not establish the connection. For details on using the Subsystem Startup Service exit routine, see *IMS Version 15 Exit Routines*.

Related concepts

“Establishing dependent region connections” on page 132
Connections can be established to MPP, IFP, or BMP regions.

Establishing dependent region connections

Connections can be established to MPP, IFP, or BMP regions.

MPP and IFP regions

Identify processing for an MPP or an IFP dependent region is similar to identify processing for the control region in that the dependent region automatically activates its Identify exit routine to establish a connection during dependent region initialization, unless the external subsystem defers the identify.

If the control region connection has not been established when the dependent region would automatically identify, IMS attempts to identify the control region. If successful, the dependent region identify is performed. Thus, if the control region identify is deferred but dependent regions are allowed to connect automatically, the control region Identify exit routine might be activated (automatically) when a dependent region is started.

When the identify for a dependent region is deferred, the connection to the external subsystem is established when the first application program call to the external subsystem is issued in the region. In this case, connection processing is the same as for a BMP dependent region.

BMP regions

The connection from a BMP dependent region is not established until the first application call to the external subsystem is processed by the region. The connection is automatically established.

Return code 4 from the Initialization exit routine (deferred identify) for a BMP region has no effect.

Notify message

IMS passes the address of a notify message on the Identify exit routine invocation for the control region.

If the external subsystem is not active (has not been started), the Identify exit routine can indicate (return code 4) to IMS that the notify message has been accepted and will be sent to IMS when the external subsystem is active. The external subsystem, once started, sends the message to IMS using an internal z/OS **MODIFY** command (SVC 34) to alert IMS that it is ready to connect. On receipt of the notify message, IMS reactivates the Identify exit routine to establish the connection.

External subsystem code that is always present in the z/OS system (*early code*), for example, might be used as the means to pass the notify message to the external subsystem. The Identify Exit queues the message to the early code so that it is available to the external subsystem whenever it is started.

IMS passes the notify message to the Identify exit routine in the format shown in the following figure.

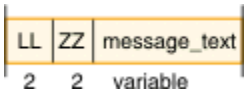


Figure 17. Notify message format

where:

LL
Is a 2-byte field containing the message length (LL + ZZ + MESSAGE_TEXT).

ZZ
Is a 2-byte field containing binary zeroes.

MESSAGE_TEXT
Is the notify message text that IMS expects to receive by the **MODIFY command** . The message text must not be altered.

Issue the **MODIFY** (F) command as follows:

```
MODIFY ims_jobname,message_text
```

The external subsystem must prefix the notify message text passed by IMS with **MODIFY** *ims_jobname*(or F *ims_jobname*), before sending the message. The following figure shows the format for the SVC 34 command input.

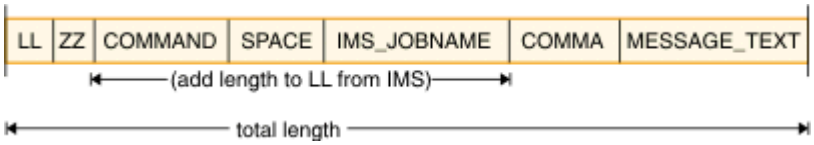


Figure 18. SVC 34 command input format

where:

LL
Is a 2-byte field containing the total length of the command input area (COMMAND + SPACE + IMS_JOBNAME + COMMA added to the length in the LL field passed to the Identify exit routine).

ZZ
Is a 2-byte field containing binary zeroes.

COMMAND
Contains the **MODIFY** command verb (C'MODIFY' or C'F').

SPACE
Is a 1-byte field containing a blank (C' ').

IMS_JOBNAME
Is an 8-byte field containing the IMS control region job name left justified and padded with blanks on the right. The Identify exit routine can obtain the job name from the TIOT pointed to by the current TCB.

COMMA
Is a 1-byte field containing a comma (C',').

MESSAGE_TEXT
Is the notify message text passed to the Identify exit routine.

Application program request support

Application calls are passed to IMS from the language interface module that you bind with the application program. The language interface branches to the appropriate IMS program request handler passing the application program call parameter list.

For calls directed to external subsystems, the language interface must also pass an external subsystem parameter list which it constructs. The purpose of this parameter list is to pass the LIT (language interface token) for the external subsystem to which the call is directed. IMS routes the application call to the external subsystem whose LIT value matches the LIT value passed on the call.

IMS passes both the call parameter list and the external subsystem parameter list to the Normal Call exit routine for the intended external subsystem. The first word of the external subsystem parameter list contains the address of a 4-byte field containing the LIT value in character format, left justified and padded on the right with blanks. IMS prescribes only the first word in the external subsystem parameter list (address of the LIT). The parameter list can be extended to provide external subsystem-dependent information to the Normal Call exit routine.

Language interface definition

IMS provides a language interface module, DFSLI000, which supports the value of SYS1. The installation can use this module or it can define its own language interface if it wants to use a LIT value other than SYS1.

When two or more external subsystems are accessed by the IMS system, the installation must define its own language interface modules because each subsystem has a unique LIT.

IMS provides the DFSLI macro to assist the installation in generating a language interface module. The code necessary to perform the language interface function is generated in the DFSLI macro expansion. The IMS macro library must be supplied when the macro statements are compiled to generate the module.

Related concepts
Defining the language interface module (System Definition)

Language interface entry points unique to external subsystems

The IMS language interface module provides three entry points that application calls directed to an external subsystem can exclusively use.

Two of the entry points are associated with an implied LIT value specified with the DFSLI macro. (The language interface module, generated by the DFSLI macro, contains the specified LIT value as a hard-coded constant.) The third entry point is not associated with an implied LIT value; it allows the application program to specify the LIT value when it makes the application call. For all entry points, register 1 contains the address of the parameter list which IMS passes to the external subsystem. The following are language interface entry points to external subsystems:

DSNHLI
Entry point associated with an implied LIT value.

The application program does not need to know which subsystem provides access to the external resources it uses. (If the external subsystem is Db2 for z/OS, this entry point is used for SQL calls.)

DSNWLI
Entry point associated with an implied LIT value.

The application program does not need to know which subsystem provides access to the external resources it uses. (If the external subsystem is Db2 for z/OS, this entry point is used for Instrument Facility calls.)

DFSESS
Entry point allowing an application program to specify an LIT value.

The application program must know which subsystem provides access to the external resources it uses. The application program must specify the address of the LIT value as the first parameter in the application call list. Before it passes control to the external subsystem, IMS increments the address of the application call list by four to skip over the LIT value parameter.

Restriction: Do not use the DFSESS entry point to communicate with a Db2 for z/OS subsystem.

Accessing multiple external subsystems

An application program can access DL/I and an external subsystem in the same execution. Whether or not an application program can access more than one external subsystem in the same execution can be restricted by the language interface.

Where the data (call) interface provided to application programs by one external subsystem (product) is distinct from the interface provided by another external subsystem (for example, DL/I calls as distinct from SQL calls), an application can access both subsystems because the language interface paths can be different. Where the data interface is the same, as in the case of two external subsystems of the same type (two instances of the external subsystem) or two external subsystem products that use the same call interface (for example, SQL), an application cannot access both in the same execution unless the application is written to be dependent on data location. (The dependency is intrinsic in the case of different call interfaces.)

Resource recovery token

A 16-byte recovery token is used to uniquely identify a unit of work across all subsystems to which the application has thread connections. IMS passes the token to the Signon exit routine before the thread is created.

For commit and resolve indoubt processing, IMS passes the recovery token to identify the unit of work for which the requested action is to be taken.

The recovery token is constructed as shown in the following figure.

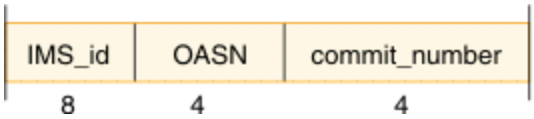


Figure 19. Recovery token format

where:

IMS-id
Is the IMS system ID (1 to 4 characters), left justified and padded with blanks on the right to eight bytes.

OASN
Is a 4-byte binary origin application sequence number assigned to the application when it is scheduled. The OASN is assigned based on the scheduling order within the IMS system since the last cold start. It is also referred to as the application schedule number.

commit_number
Is a 4-byte binary commit number. The commit number is initialized to binary zeroes when the application is scheduled and then incremented after each commit is processed for the application.

The external subsystem should check the recovery token passed at signon for uniqueness. Cold starts of IMS can cause a recovery token to be generated that is a duplicate of a recovery token that is indoubt in the external subsystem. The Signon exit routine can indicate to IMS that the recovery token passed was found to be a duplicate, in which case IMS terminates the application program with an abend. The Commit Prepare exit routine can also indicate that the token is a duplicate that supports external subsystems that choose not to associate the recovery token with the unit of work until commit is processed.

The installation uses the **/DISPLAY SUBSYS** command with the OASN keyword to determine what units of work are in indoubt status in IMS. The installation can use the **/CHANGE** command (when necessary) to manually delete indoubt units of work in IMS. The **/CHANGE** command only affects unit of work status in IMS. There is no communication with the external subsystem. These commands use only the OASN and not the full recovery token; **/DISPLAY** lists only the OASN portion of the recovery token (in decimal format) and **/CHANGE** accepts just the OASN (again in decimal format). (Within IMS, the OASN is unique across all known units of work.)

Terminating the external subsystem connection

IMS terminates the external subsystem connection (control region connection) in an orderly manner in a number of different circumstances.

IMS terminates the external subsystem connection (control region connection) in an orderly manner when one of the following occurs:

- IMS processes a **/STOP SUBSYS** command.
- The external subsystem (ESAP) activates the IMS subsystem termination service exit routine.
- The external subsystem posts the termination ECB provided on the Identify exit routine invocation.
- Certain attach processing errors are encountered.
- IMS is shutting down.

IMS allows existing threads to the external subsystem to complete processing. When all threads have terminated, IMS terminates the connection by activating the Terminate Identify exit routine from the control region.

When the connection is terminated due to a **/STOP SUBSYS** command, the termination ECB being posted, or processing errors, IMS puts the external subsystem connection in stopped status. Once in stopped status, IMS does not allow a connection to be reestablished. A **/START SUBSYS** command is required to remove the stopped status.

Termination requested by the external subsystem

The external subsystem can cause the connection to be terminated either by posting the termination ECB or by activating the Subsystem Termination Service exit routine from the ESAP. The connection is not put in stopped status when the service is used.

The termination service might be used in conjunction with the external subsystem command exit function. For example, when the command exit routine is activated with an external subsystem termination command supplied on an IMS **/SSR** command, the exit routine could activate the Subsystem Termination Service exit routine to cause the connection to the external subsystem to be terminated.

On the initial identify performed in the control region, IMS provides the external subsystem with the address of a termination notification ECB. When the subsystem needs to terminate the connection, it posts the ECB. The ECB is located in CSA. Depending on the post code, IMS terminates the connection in the following manner:

Deactivate all active threads, prohibit the initiation of any new threads, and then terminate the connection. Upon completion of the terminate function, the connection is set in a stopped state.

Supported Post Codes:

X'40000000'
Reserved for IMS.

X'40000008'
External subsystem is terminating in an orderly fashion.

X'4000xxxx'
All other post codes are interpreted as a quick or catastrophic shutdown of the external subsystem.

Related reference
Subsystem Termination Service exit routine (Exit Routines)

Dependent region connections

At dependent region termination, the Signoff and Terminate Identify exit routines are not activated unless the control region connection is to be terminated.

That is, the dependent region Signoff and Terminate Identify exit routines are activated only when the IMS system continues to run without a connection to the external subsystem, such as when either the external subsystem has posted the termination ECB provided on the identify, the ESAP has activated the IMS Subsystem Termination Service, or IMS has processed a **/STOP SUBSYS** command. Otherwise, IMS does not communicate dependent region termination to the external subsystem.

IMS expects the external subsystem to monitor the dependent TCB with a z/OS End Of Task (EOT) exit routine. The subsystem should do any signoff and terminate identify processing it requires when the EOT exit routine is notified of the region termination.

The external subsystem must also monitor EOT exit routine for dependent regions for which a thread was created. When an application program terminates abnormally, the Terminate Thread exit routine is not called.

When the control region connection is to be broken, a signoff followed by a terminate identify is done for the dependent region at region termination or after thread termination if a thread was active when the request to break the system connection was received. The Signoff exit routine is called only once even though more than one signon might have been done for the region. IMS continues its signoff and terminate identify processing and does not reactivate these exit routines if they encounter errors.

Explanation of stopped status

The installation should be aware of the conditions that cause IMS to stop the external subsystem connection.

After an external subsystem connection is stopped, the **/START SUBSYS** command must be used to reestablish the connection. Stopped status is carried across restarts. The following list includes the conditions that cause the stopped status to be set:

- When the external subsystem posts the subsystem termination ECB provided during identify processing. Regardless of the post code type (for example, orderly or catastrophic), the connection is stopped upon completion of the termination processing.
- On abnormal termination of the IMS task (TCB) in the control region under which the ESAP is activated, the external subsystem connection is marked stopped.

  If the control region abends, it is unlikely the stopped state will be set. After a successful IMS restart, the connection is in the state it was prior to the abend.
- The obvious case is after processing of the **/STOP** command. Even if IMS abends while processing the **/STOP** command, the stopped state is set.
- When IMS restarts, if outstanding recovery tokens exist for an external subsystem that is no longer defined to IMS (for example, the installation deleted its definition from the external subsystem definition member in IMS.PROCLIB), stopped status is set for that external subsystem.

Part 5. IMS Connect and TCP/IP communications

The IMS Connect function of IMS provides access to both IMS DB and IMS TM from TCP/IP-enabled environments.

© Copyright IBM Corp. 1974, 2020

Chapter 10. Overview of IMS Connect

IMS Connect provides high performance TCP/IP communications between one or more IMS Connect clients and one or more IMS systems. IMS Connect supports both IMS DB and IMS TM systems.

IMS Connect enables:

- ISC users to link with IBM CICS Transaction Server for z/OS over a TCP/IP connection.
- MSC and OTMA users to send messages from one IMS system to another by using IMS-to-IMS TCP/IP connections.
- Distributed clients to exchange messages with IMS DB by using TCP/IP connections and the Open Database Manager (ODBM) component of the IMS Common Service Layer (CSL).
- Distributed clients to exchange messages with IMS TM by using TCP/IP connections and OTMA.
- IMS Operators that use IBM Management Console for IMS and Db2 for z/OS to issue commands to an IMSplex and receive command replies by using TCP/IP and the IMS Operations Manager (OM).

IMS Connect provides the following features:

- Commands to manage the communication environment.
- Assistance with workload balancing.
- Reduced design and coding efforts for client applications.
- Support for IMSplexes, which enables communications with other IMSplex members, such as:
  - IMS for Intersystem Communication (ISC) and Multiple Systems Coupling (MSC) support
  - IMS for Multiple Systems Coupling (MSC) support
  - ODBM
  - OM
- Connectivity between IMS and CICS on ISC TCP/IP links.
- Connectivity between IMS Connect instances, which supports IMS communications components, such as MSC and OTMA.
- TCP/IP access to IMS application programs and operations on demand with advanced security and transactional integrity.
- TCP/IP access for distributed applications to IMS databases on demand through clients such as the IMS Universal drivers and ODBM.
- XML conversion support for certain IMS Connect clients, such as IMS Enterprise Suite SOAP Gateway and IMS Web 2.0 Solution for IBM Mashup Center. The IMS Connect XML conversion support converts input messages into the data structures expected by IMS application programs written in select programming languages, thereby eliminating the need to create or modify IMS application programs to process XML.

IMS Connect connects to IMS DB through ODBM for direct access to databases that are managed by IMS DB. IMS Connect connects to IMS TM for transaction processing support through Open Transaction Manager Access (OTMA).

Communications between IMS Connect and other IMSplex members, such as IMS for MSC support, ODBM, and OM, requires the use of the IMS Structured Call Interface (SCI).

IMS Connect performs router functions between its clients and IMS and IMSplex resources. Request messages that are received from distributed clients through TCP/IP connections are passed to an IMS system, referred to as a data store, through z/OS cross-system coupling facility (XCF) sessions. IMS Connect receives response messages from the data store and then returns them to the originating TCP/IP.

If the data store terminates, the status of the data store is sent to IMS Connectfrom OTMA through XCF. If IMS Connect was connected to the data store when the data store terminated, when the data store

© Copyright IBM Corp. 1974, 2020

is restarted, IMS Connect is notified and automatically reconnects to the data store. You do not need to manually reconnect to the data store.

Generally, the term *data store* refers to an IMS system. More precisely, however, the term data store represents the OTMA target member (tmember) connection to an IMS system. For example, an instance of IMS Connect can have multiple data store definitions for the same IMS system, in which case each data store represents a different OTMA tmember connection to that IMS system.

IMS Connect also supports callout requests from IMS application programs running in IMS dependent regions. IMS application programs issue callout requests to request data or services from a provider that is external to the IMS installation. During a callout request, IMS acts as a client and the external provider is the server.

Related concepts

IMS Connect definition and tailoring (System Definition)

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

"IMS Connect support for callout requests" on page 201

IMS Connect is a required component when IMS application programs issue callout requests through OTMA to data or service providers that are external to the IMS installation. For both types of callout request, IMS Connect serves as the TCP/IP gateway between the IMS Connect client that use TCP/IP and the OTMA component of IMS.

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

IMS Connect commands (Commands)

IMS Connect exit routines (Exit Routines)

HWS messages (IMS Connect) (Messages and Codes)

IMS Connect return and reason codes (Messages and Codes)

IMS Connect client support

As a TCP/IP server and a message router for IMS, IMS Connect provides access to IMS TM, IMS DB, and the CSL Operations Manager (OM). The client support provided by IMS Connect differs, depending on which type of access the IMS Connect client needs.

IMS Connect supports TCP/IP clients that communicate with socket calls, as well as TCP/IP clients that communicate with different input data stream formats.

IMS DB client support

For access to IMS DB, IMS Connect, with the CSL Open Database Manager (ODBM), supports the following types of clients:

- Application programs that use the IMS Universal Database resource adapter for the Java EE platform
- Application programs that use the IMS Universal JDBC driver
- Application programs that use the IMS Universal DL/I driver
- User-written client application programs that use the open standard DRDA communications architecture

The following figure illustrates an IMS Connect system configuration that supports IMS DB client communications:

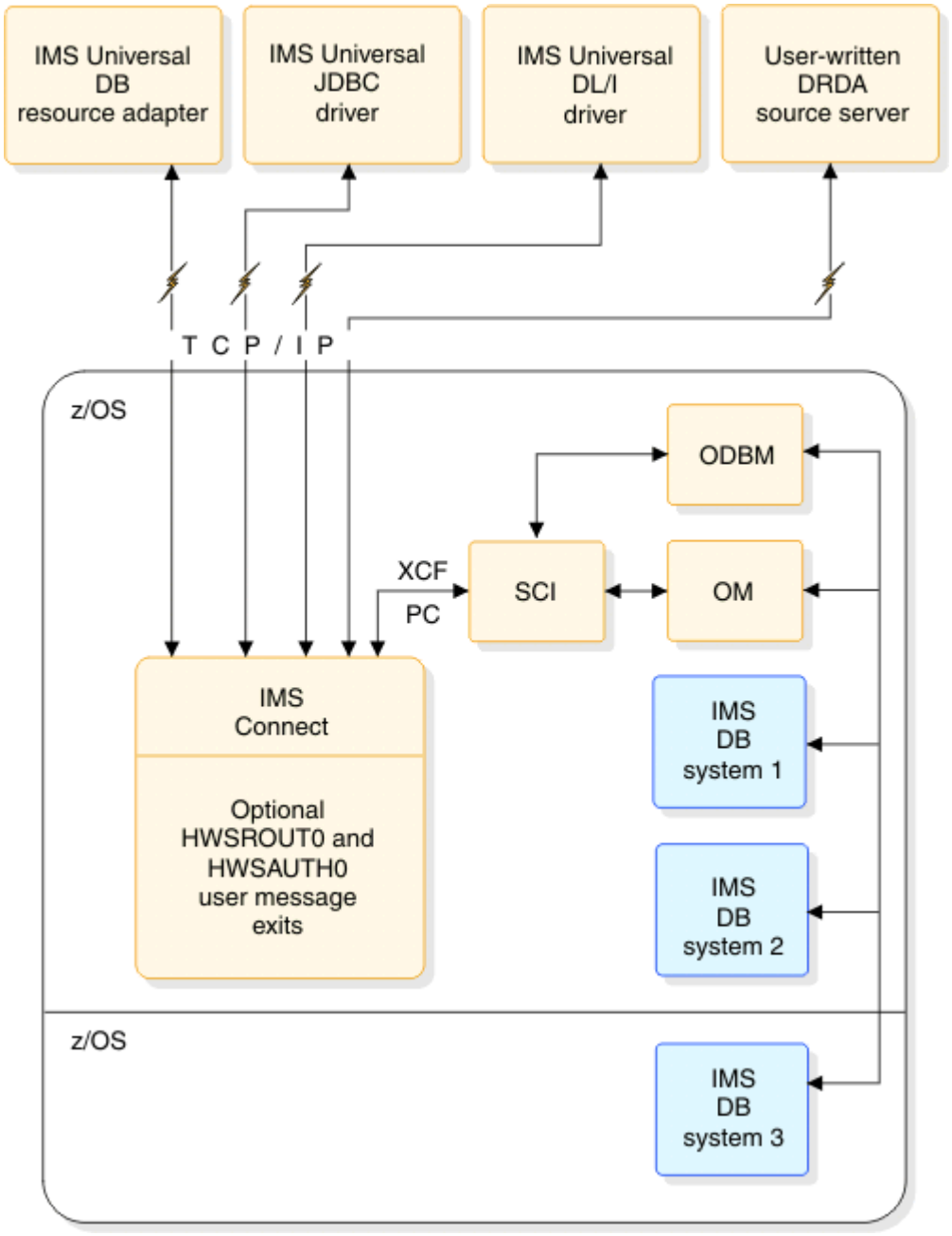


Figure 20. Overview of IMS Connect support for IMS DB systems

IMS TM client support

For access to IMS TM, IMS Connect supports the following types of clients:

- User-written TCP/IP client application programs that use the IMS request message (IRM) header to communicate protocol options to IMS Connect
- IMS TM Resource Adapter (previously known as IMS Connector for Java)
- IMS Enterprise Suite SOAP Gateway, which includes XML message conversion support
- IMS Web 2.0 Solution for IBM Mashup Center, which includes XML message conversion support

The following figure illustrates an IMS Connect system configuration that supports IMS TM client communications:

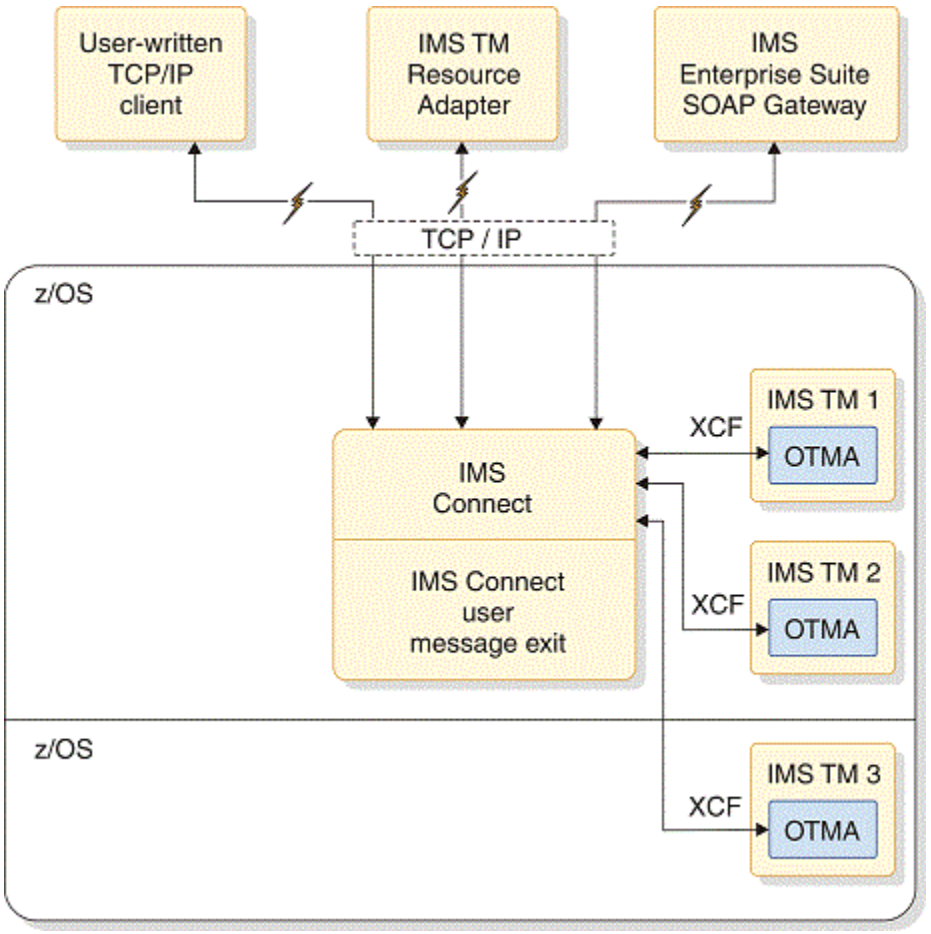


Figure 21. Overview of IMS Connect support for IMS TM systems

For IMS TM IMS Connect clients, you can write user-message exits that execute in the IMS Connect address space to convert the format of client input messages to the OTMA message format before IMS Connect sends the message to IMS. The user-written message exits also convert the OTMA message format to the customer message format before returning the message to IMS Connect. IMS Connect then sends output to the client.

IMS Connect supports TCP/IP communications between IMS TM and distributed Java applications through the IMS TM Resource Adapter running under either WebSphere Application Server for distributed platforms.

For IMS TM, IMS Connect also supports the SOAP Gateway, which is a web services solution that enables IMS applications to inter-operate outside of the IMS environment through SOAP to provide and request services that are independent of platform, environment, application language, or programming model.

IMS Connect supports the IMS Web 2.0 Solution for IBM Mashup Center, which enables the integration of existing IMS assets into Web 2.0 mashup and application solutions, providing users access to IMS transactions through RSS, ATOM, or XML feed.

For SOAP Gateway and IMS Web 2.0 Solution clients, IMS Connect also provides XML conversion support, which converts incoming XML messages into the data structures of some of the common programming languages supported by IMS application programs.

IMSplex operations support

To issue IMS type-2 commands to the CSL OM and receive command responses through a TCP/IP connection, IMS Connect supports clients such as IBM Management Console for IMS and Db2 for z/OS. A single IMS Connect can support communication between the TCP/IP client and any IMS within an IMSplex.

The following figure illustrates an IMS Connect system configuration that supports IMS TM client communications:

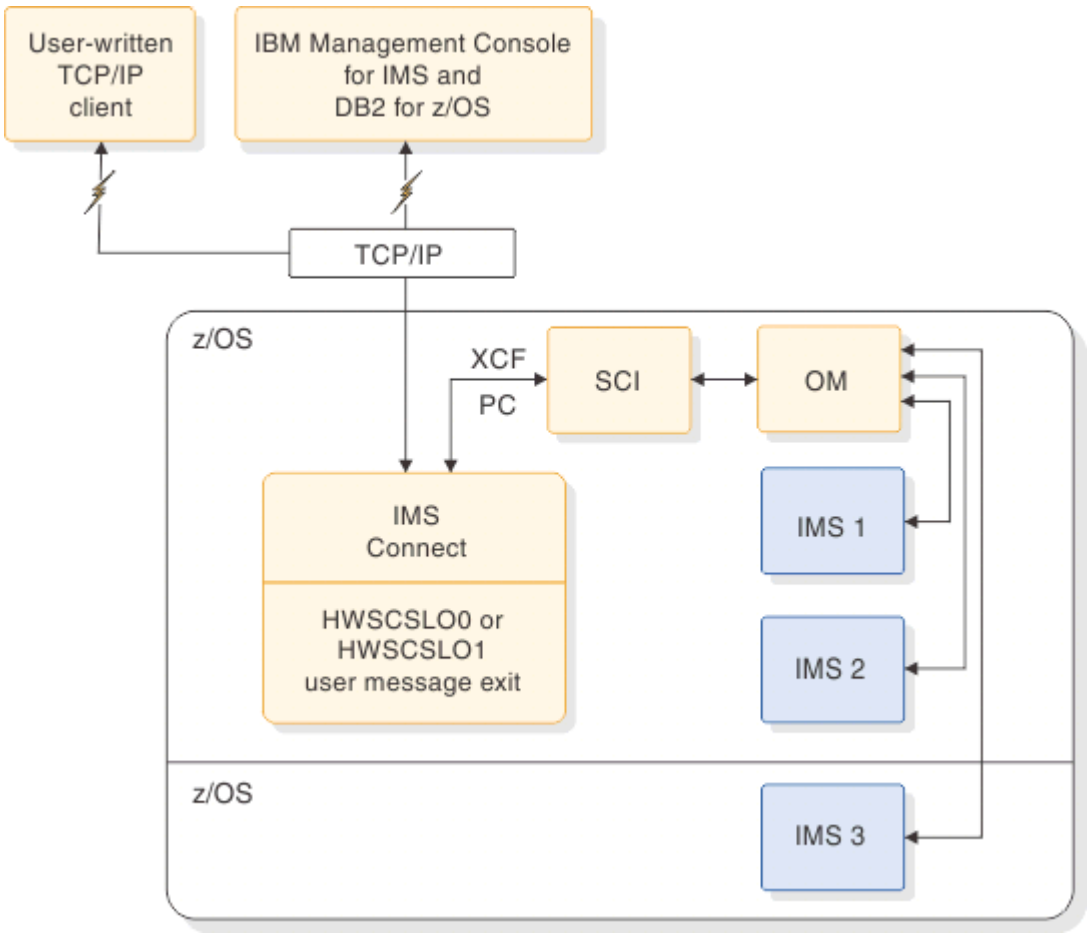


Figure 22. Overview of IMS Connect support for TCP/IP clients

IMS Connect support for access to IMS DB

For IMS Connect clients, such as the IMS Universal drivers, that access databases that are managed by IMS DB in DBCTL and DB/TM environment, IMS Connect manages TCP/IP connections and routes incoming access requests among the instances of the CSL Open Database Manager (ODBM) and the IMS DB systems in an IMSplex.

IMS Connect is the TCP/IP server and front-end IMSplex message router for the IMS Universal drivers, which include:

- IMS Universal Database resource adapter for the Java EE platform
- IMS Universal JDBC driver
- IMS Universal DL/I driver

The IMS Universal drivers provide direct, non-transactional access to IMS databases through TCP/IP connections for distributed application programs or local z/OS application programs.

The IMS Universal drivers simplify the development of IMS Connect client application programs that access IMS DB through TCP/IP connections. The IMS Universal drivers do not use the IMS Connect IMS Request Message (IRM) communications protocol. The IMS Universal drivers also shield application developers from the underlying Distributed Relational Database Architecture™ (DRDA) that is used internally for communications between the IMS Universal drivers and IMS Connect.

Because IMS Connect supports the DRDA protocol and, with ODBM, is a complete DRDA target server, you can write application programs to the DRDA protocol directly; however, the IMS Universal Database

resource adapter for the Java EE platform is the recommended API for accessing IMS databases through TCP/IP from a distributed environment.

IMS Connect support for the IMS Universal drivers includes support for global two-phase commit transactions.

IMS Connect supports communication with the IMS Universal drivers only on dedicated DRDA ports and only through shareable persistent sockets.

IMS Connect security support includes the IMS Connect DB Security user exit routine (HWSAUTH0), which can be used for greater control over the authentication of user IDs on connections that access IMS DB. RACF is also supported.

IMS Connect support for the IMS Universal drivers is defined by the ODACCESS configuration statement in the IMS Connect configuration PROCLIB member and requires at least one instance of ODBM running in the same IMSplex as IMS Connect.

Connection routing for IMS Connect clients that connect to IMS DB

IMS Connect routes incoming connections from client applications that are connecting to IMS DB based on an *alias name* and the instance of ODBM to which the alias name belongs. The client application programs can specify an alias name on their incoming request messages.

IMS Connect keeps track of which alias names belong to which instance of ODBM in a internal tracking table that is populated during registration with each instance of ODBM. Information about the ODBM instances and alias names is also stored in an ODBM list in an exit interface block mapped by the HWSXIBOD macro. The address of the ODBM list is stored in a table mapped by the HWSXIB macro.

IMS Connect also provides the IMS Connect DB Routing user exit routine (HWSROUT0) that you can use to further control the routing of incoming message from clients that access IMS DB.

If IMS Connect receives a blank alias name from the client application or the HWSROUT0 exit routine, IMS Connect routes incoming connections in round-robin fashion among all of the instances of ODBM known to IMS Connect. If IMS Connect receives from the client application or the HWSROUT0 exit routine an alias name that is shared by multiple instances of ODBM, IMS Connect routes the incoming connections that specify that alias name in round-robin fashion among all of the instances of ODBM that share the alias name.

IMS Connect support for the IMS TM Resource Adapter

To support Java application programs connecting to IMS TM, IMS Connect supports the IMS TM Resource Adapter.

The IMS TM Resource Adapter runs under WebSphere Application Server in distributed environments and is used by Java applications, Java Platform, Enterprise Edition (Java EE) applications or web services to access IMS transactions that are running on host IMS systems. It is also used by IMS applications that run in IMS dependent regions to make asynchronous callout requests to external Java EE applications.

The IMS TM Resource Adapter connects to IMS Connect through the TCP/IP communications protocol.

When a Java application submits a transaction request to IMS through the IMS TM Resource Adapter, IMS Connect sends the transaction request to IMS Open Transaction Manager Access (OTMA) by using the z/OS cross-system coupling facility, and the transaction runs in IMS. The response is returned to the Java application using the same path.

When an IMS application invokes an external enterprise JavaBeans (EJB) component or web service through a callout request, IMS Connect retrieves the callout request from a hold queue and passes it to the IMS TM Resource Adapter, which in turn passes the request to an EJB application in WebSphere Application Server that is set up to receive callout requests. The EJB starts a connection to IMS Connect via the IMS TM Resource Adapter. The IMS TM Resource Adapter polls IMS Connect to retrieve the callout requests from the hold queue. The EJB processes the request, and returns any response data to IMS by issuing a normal IMS transaction request.

IMS Connect identifies TCP/IP connections by a client ID that is submitted by the IMS TM Resource Adapter.

For connections on persistent TCP/IP sockets, IMS Connect can generate a client ID for the IMS TM Resource Adapter in any of the following cases:

- The IMS TM Resource Adapter requests that IMS Connect generate a unique client ID in the event that a client ID submitted by the IMS TM Resource Adapter is a duplicate of an existing IMS TM Resource Adapter client ID.
- The IMS TM Resource Adapter submits a blank client ID.

IMS Connect support for command requests to Operations Manager (OM)

IMS Connect (ICON) clients can send command requests to the Operations Manager (OM) to be processed by components in the IMSplex such as IMS, ODBM, and ICON.

Sending a command request to OM is similar to sending a transaction to IMS. ICON clients build the command request using the IMS request message (IRM) header format.

The following are command-related settings that user-written ICON clients must set in the command request:

- Set the user message exit IRM_ID field to HWSCSLO0 or HWSCSLO1 string values ‘*HWSCSL*’ or ‘*HWSCS1*’.
- Set the destination IRM_IMSDESTID to the IMSplex name connected to the ICON. The IMSplex name can be found in the TMEMBER= parameter of the IMSPLEX statement in the ICON HWSCFGxx configuration member.
- Set the application data to the command string 'CMD(*command string*)' such as 'CMD(QRY IMSCON TYPE(PORT) NAME(*) SHOW(STATUS))'.

ICON API for Java clients can issue the following method calls by setting the following fields:

- Set the user message exit by issuing TmInteraction.setImsConnectUserMessageExitIdentifier(exit name). Exit name can be either TmInteraction.IMS_CONNECT_USER_MESSAGE_EXIT_IDENTIFIER_FOR_HWSCSLO0 or TmInteraction.IMS_CONNECT_USER_MESSAGE_EXIT_IDENTIFIER_FOR_HWSCSLO1.
- Set the destination by issuing TmInteraction.setImsDatastoreName(IMSplex name).
- Set Interaction type to command by issuing TmInteraction.setInteractionTypeDescription(TmInteraction.INTERACTION_TYPE_DESC_TYPE2_COMMAND).
- Set the application data as shown below:

```
String inputMessage = "CMD(command string)";

short inputMessageLength = (short) (inputMessage.length() + 4);

ByteBuffer inputMessageInByteFormat = ByteBuffer.allocate(inputMessageLength);

inputMessageInByteFormat.position(0);

inputMessageInByteFormat.putShort((short) inputMessageLength);

inputMessageInByteFormat.putShort((short) 0);

inputMessageInByteFormat.put(inputMessage.getBytes(Charset.forName("CP1047")));InputMessage im
= commandTi.getInputMessage();

im.setInputMessageData(inputMessageInByteFormat.array());
```

Example command request message for command QRY IMSCON TYPE(PORT) NAME(*) SHOW(STATUS) using exit HWSCSLO1 and destination PLEX1:

```
Offset 0 4 8 C EBCDIC Data
------ ---------------------------------- ------------------
```

```
+000000 000000E3 00A80500 5CC8E6E2 C3E2F15C |...T.y..*HWSCS1*|
+000010 00000000 00631000 C3D3C9C5 D5E3F0F1 |........CLIENT01|
+000020 00208040 40404040 40404040 D7D3C5E7 |...          PLEX|
+000030 F1404040 40404040 40404040 E4E2D9E3 |1            USRT|
+000040 F0F0F140 E2E8E2F1 40404040 5C5C5C5C |001 SYS1    ****|
+000050 5C5C5C5C D9C1C3C6 C1D7D5D4 40404040 |****RACFAPNM     |
+000060 40404040 40404040 40404040 40404040 |                 |
+000070 40404040 40404040 40404040 00000000 |             ....|
+000080 00000000 00000000 00000000 00000000 |................|
+000090 00000000 00000000 00000000 00000000 |................|
+0000A0 00000000 00000000 00000000 00330000 |.............3..|
+0000B0 C3D4C44D D8D9E840 C9D4E2C3 D6D540E3 |CMD(QRY IMSCON T|
+0000C0 E8D7C54D D7D6D9E3 5D40D5C1 D4C54D5C |YPE(PORT) NAME(*|
+0000D0 5D40E2C8 D6E64DE2 E3C1E3E4 E25D5D00 |) SHOW(STATUS)).|
+0000E0 040000                               |...             |
```

Example command response message:

```
Offset 0 4 8 C EBCDIC Data
------ ---------------------------------- ------------------
+000000 000010E7 4C6FA794 9340A585 99A28996 |...X<?xml versio|
+000010 957E7FF1 4BF07F6F 6E4C5AC4 D6C3E3E8 |n="1.0"?><!DOCTY|
+000020 D7C54089 94A296A4 A340E2E8 E2E3C5D4 |PE imsout SYSTEM|
+000030 407F8994 A296A4A3 4B84A384 7F6E4C89 | "imsout.dtd"><i|
+000040 94A296A4 A36E4C83 A3936E4C 96949581 |msout><ctl><omna|
+000050 94856ED6 D4F1D6D4 4040404C 61969495 |me>OM1OM   </omn|
+000060 8194856E 4C9694A5 A2956EF1 4BF84BF0 |ame><omvsn>1.8.0|
+000070 4C619694 A5A2956E 4CA79493 A5A2956E |</omvsn><xmlvsn>|
...
+000FE0 827E7FA8 85A27F40 616E4C61 83948499 |b="yes" /></cmdr|
+000FF0 A2978884 996E4C83 948499A2 978481A3 |sphdr><cmdrspdat|
+001000 816E4C99 A2976ED7 D6D9E34D F9F9F9F9 |a><rsp>PORT(9999|
+001010 40404040 5D40D4C2 D94DC8E6 E2F14040 |    ) MBR(HWS1  |
+001020 40404040 40404040 40405D40 C3C34D40 |          ) CC( |
+001030 4040F05D 40E2E3E3 4DC1C3E3 C9E5C55D |  0) STT(ACTIVE)|
+001040 404C6199 A2976E4C 99A2976E D7D6D9E3 | </rsp><rsp>PORT|
+001050 4DF9F9F9 F8404040 405D40D4 C2D94DC8 |(9998    ) MBR(H|
+001060 E6E2F140 40404040 40404040 4040405D |WS1            )|
+001070 40C3C34D 404040F0 5D40E2E3 E34DC1C3 | CC(   0) STT(AC|
+001080 E3C9E5C5 5D404C61 99A2976E 4C99A297 |TIVE) </rsp><rsp|
+001090 6ED7D6D9 E34DF5F5 F5F5C440 40405D40 |>PORT(5555D   ) |
+0010A0 D4C2D94D C8E6E2F1 40404040 40404040 |MBR(HWS1        |
+0010B0 40404040 5D40C3C3 4D404040 F05D40E2 |    ) CC(   0) S|
+0010C0 E3E34DC1 C3E3C9E5 C55D404C 6199A297 |TT(ACTIVE) </rsp|
+0010D0 6E4C6183 948499A2 978481A3 816E4C61 |></cmdrspdata></|
+0010E0 8994A296 A4A36E                      |imsout>         |
```

IMS Connect support for ISC TCP/IP communications

IMS Connect manages the TCP/IP connections and protocols for IMS when Intersystem Communication (ISC) parallel sessions use TCP/IP to link to IBM CICS Transaction Server for z/OS.

The following figure shows the basic flow for a single ISC parallel session that uses TCP/IP. In the figure, the ISC TCP/IP terminal is defined dynamically in IMS by an ETO logon descriptor.

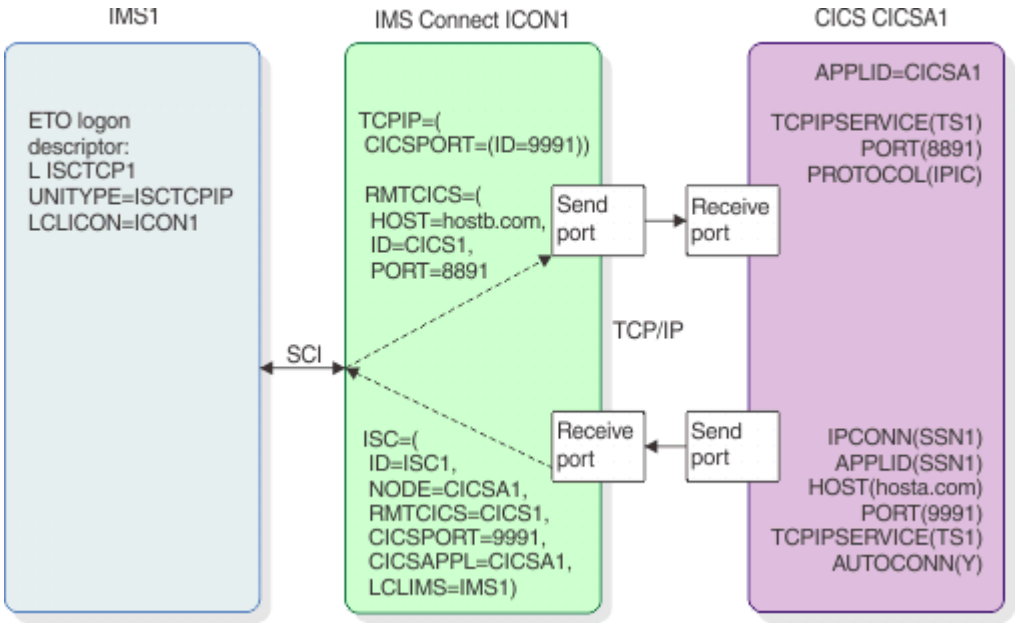


Figure 23. Overview of IMS Connect support for an ISC parallel session that uses TCP/IP

Each ISC parallel session requires two sockets in IMS Connect: a send socket and a receive socket. CICS also requires two sockets for each ISC parallel session. An ISC link can support multiple ISC parallel sessions.

The ISC and RMTCICS statements, along with the CICSPORT keyword on the TCPIP statement, together define the TCP/IP connection from IMS Connect to IMS and from IMS Connect to CICS. These statements are defined in the IMS Connect configuration member in the IMS.PROCLIB data set.

Each TCP/IP socket provides a one-way path for transactions and reply data messages between IMS Connect and CICS. Acknowledgment messages (ACKs or NAKs) are sent on the socket connection from which the transaction or reply data message was received.

Communication between IMS Connect and IMS is enabled by the Structured Call Interface (SCI) of the IMS Common Service Layer (CSL).

ISC TCP/IP communication does not support RACF PassTickets.

ISC messages between CICS and IMS cannot be modified or routed by any IMS Connect exit routines. The IMS Connect user message exit routines, such as HWSJAVA0, HWSSMPL0, or HWSSMPL1, are not used for ISC TCP/IP communication. The Port Message Edit exit routine is not supported.

ISC TCP/IP communication is supported by IMS type-2 commands. Only a limited number of IMS Connect WTOR and z/OS MODIFY commands, such as VIEWHWS and QUERY MEMBER, support ISC TCP/IP communication.

The IMS Connect Event Recorder exit routine (HWSTECL0) records events that are specific to ISC TCP/IP communications.

Related tasks

“ISC communication with CICS over TCP/IP” on page 593
TCP/IP can be used to support ISC connections between IMS and IBM CICS Transaction Server for z/OS subsystems.

Related reference

ISC statement (System Definition)
RMTCICS statement (System Definition)

IMS Connect support for IMS-to-IMS TCP/IP communications

IMS Connect manages the TCP/IP connections and protocols for IMS systems that communicate with each other across a TCP/IP network.

An IMS-to-IMS TCP/IP connection typically links two IMS systems that are installed at different locations. Each IMS system communicates with an IMS Connect instance at the same location. The TCP/IP connection that the two IMS system use to communicate with each other is established by the IMS Connect instances at each location.

The following figure illustrates possible configurations for IMS-to-IMS TCP/IP connections.

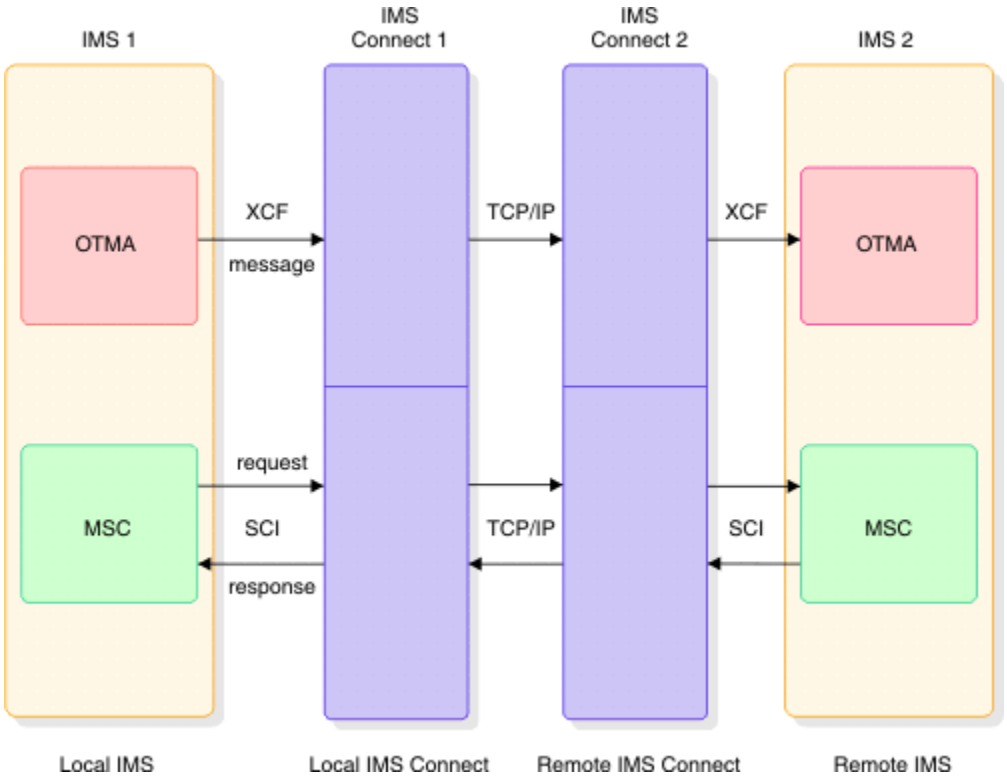


Figure 24. Example configuration for IMS-to-IMS TCP/IP communications

The RMTIMSCON statement in the IMS Connect configuration member in the IMS.PROCLIB data set defines the TCP/IP connection from one IMS Connect instance to the other. Depending on which IMS communications component uses the TCP/IP connection, additional configuration statements, descriptors, or system definition macros are required to complete the communications path between the two IMS systems.

Each defined TCP/IP connection provides a one-way path for messages from one IMS system to flow to the other. For messages to flow back in the other direction, a second TCP/IP connection must be defined in the reverse direction. Acknowledgment messages (ACKs or NAKs) are the exception to this rule.

Optionally, IMS Connect can secure the IMS-to-IMS TCP/IP connections by using RACF PassTickets. When PassTicket security is enabled, the sending IMS Connect instance generates a PassTicket and the receiving IMS Connect instance verifies the PassTicket with RACF. After the PassTicket is verified, the receiving IMS Connect classifies the connection as coming from a trusted user. As long as the connection persists, no further security checking is performed.

The IMS Connect user message exit routines, such as HWSJAVA0, HWSSMPL0, or HWSSMPL1, are not used for IMS-to-IMS TCP/IP communications.

IMS-to-IMS TCP/IP communications are supported by the following IMS Connect command formats:

- WTOR
- z/OS MODIFY

- IMS type-2

The IMS Connect Event Recorder exit routine (HWSTECL0) records events that are specific to IMS-to-IMS TCP/IP communications.

The IMS communications components that use IMS-to-IMS TCP/IP communications include Multiple Systems Coupling (MSC) and Open Transaction Manager Access (OTMA).

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

"Securing IMS-to-IMS TCP/IP connections" on page 181

To secure IMS-to-IMS TCP/IP connections, IMS Connect uses RACF PassTickets to establish one instance of IMS Connect as a trusted user of another instance of IMS Connect.

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

IMS Connect commands (Commands)

QUERY IMSCON commands (Commands)

UPDATE IMSCON commands (Commands)

IMS Connect Event Recorder exit routine (HWSTECL0) (Exit Routines)

MSC and IMS-to-IMS TCP/IP communications

For MSC, the TCP/IP connections between two IMS Connect instances complete a path for an MSC physical link. One or more MSC logical links can be assigned to the MSC physical link.

Messages on MSC links travel in both directions. For each MSC logical link, two socket connections are opened: a send socket and a receive socket. To support MSC links, you must define a pair of corresponding TCP/IP connections, one in each IMS Connect instance on each side of the IMS-to-IMS TCP/IP connection.

To define a complete IMS-to-IMS TCP/IP communications path for MSC, you must code the following items in the IMS Connect and IMS instances at each side of the connection:

- In IMS Connect, the IMS Connect configuration statements, including the RMTIMSCON statement, which is required for IMS-to-IMS TCP/IP communications, and the MSC statement, which is required for communication between IMS Connect and MSC.
- In IMS, the MSC system definition macros or type-2 CREATE commands that are required to define all MSC link types. The MSPLINK macro and the CREATE MSPLINK command have parameters that are specific to the TCP/IP MSC physical link type.

A simplified IMSplex configuration is required to support IMS-to-IMS TCP/IP communications for MSC. The communications path between IMS Connect and MSC is managed by the Structured Call Interface (SCI) component of the IMSplex. IMS type-2 command support for IMS Connect and MSC also requires the Operations Manager (OM) component of the IMSplex.

For MSC links that use TCP/IP generic resources, IMS Connect provides routing and affinity management support.

Security for the TCP/IP connection can be implemented by using the optional RACF PassTicket support. Transaction authorization can also be implemented in the IMS system.

In each IMS system, a TCP/IP MSC link is operationally like a VTAM MSC link. An MSC physical link can be started by issuing either the IMS type-1 command **/RSTART LINK** or the IMS type-2 command **UPDATE MSLINK** START(COMM) in either one of the linked IMS systems.

In IMS Connect, after a link has been started, you can monitor, stop, and restart the MSC links and their associated socket connections by using IMS Connect WTOR, z/OS MODIFY, or IMS type-2 commands. In IMS, most definition, operations, and administrative tasks and processes are the same for the MSC TCP/IP link type as they are for the MSC VTAM link type.

Related concepts

"Overview of Multiple Systems Coupling" on page 667

Multiple Systems Coupling (MSC) makes it possible for transactions to be entered in one IMS and processed in another IMS.

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

Defining a TCP/IP generic resource group for MSC (System Definition)

Related reference

CREATE MSPLINK command (Commands)

IMS Connect, MSC, and TCP/IP generic resources

IMS Connect provides routing and connection management support for MSC links that connect to a TCP/IP generic resource group.

TCP/IP generic resources enable MSC links to be switched between participating IMS systems in an IMSplex without requiring system definition changes to the MSC partner systems outside of the IMSplex.

The IMS systems that participate in a TCP/IP generic resource group all specify a shared generic IMS ID in their respective IMS DFSDCxxx PROCLIB members. In IMS Connect, this generic IMS ID is specified on the GENIMSID parameter of the MSC configuration statement for each IMS system.

When IMS Connect receives the first MSC logical link request on a physical link for the TCP/IP generic resource group, IMS Connect broadcasts the request to all participating IMS systems in the group. IMS Connect routes the link request to the first IMS system to respond to IMS Connect.

When an IMS system accepts the link request, affinity is established between the IMS system and all logical links that use the physical link to the IMS system. Affinity persists for all logical links on a given MSC physical link until all logical links on the physical link terminate normally or the physical link path is stopped in IMS Connect. After the last logical link terminates and the affinity is released, the next link request on the physical link can establish affinity with a different IMS system.

You can display affinity information for a logical link in both IMS and IMS Connect. In IMS Connect, you can display affinity information by specifying any one of the following IMS Connect commands:

- The IMS type-2 format command **QUERY IMSCON** TYPE(MSC) NAME*(lclPlkid*)
- The IMS type-2 format command **QUERY IMSCON** TYPE(LINK) NAME*(lclPlkid*)
- The WTOR format command **VIEWMSC** *lclPlkid*
- The z/OS MODIFY command **QUERY MSC** NAME(*lclPlkid*)

You can control which IMS system in a TCP/IP generic resource group accepts the first logical link request on a physical link. The easiest way to do this is to start the link from the IMS system in the generic resource group.

If the link must be started from the remote IMS system, you can control which IMS system accepts a link from either the TCP/IP generic resource group or IMS Connect. In the TCP/IP generic resource group, you can stop logons to the generic IMS ID in every IMS system in the generic resource group, except in the one that requires affinity. Alternatively, in IMS Connect you can stop all MSC physical link paths except the path to the IMS system that requires affinity.

To stop a physical link path in IMS Connect, issue any one of the following IMS Connect commands:

- The IMS type-2 format command **UPDATE IMSCON** TYPE(MSC) NAME*(lclPlkid*) STOP(COMM)
- The WTOR format command **STOPMSC** *lclPlkid*
- The z/OS MODIFY command **UPDATE MSC** NAME(*lclPlkid*) STOP(COMM)

Related tasks

“Managing MSC links in a TCP/IP generic resource group” on page 726
When MSC is used with TCP/IP generic resources, each link that connects to the TCP/IP generic resource group has affinity to a specific IMS system in the group.

Defining a TCP/IP generic resource group for MSC (System Definition)

Related reference

MSC statement (System Definition)
/DISPLAY AFFIN command (Commands)
QUERY MSLINK command (Commands)
QUERY IMSCON TYPE(LINK) command (Commands)
QUERY IMSCON TYPE(MSC) command (Commands)
VIEWMSC command (Commands)
IMS Connect QUERY MSC command (Commands)

OTMA and IMS-to-IMS TCP/IP communications

For OTMA, a TCP/IP connection between two IMS Connect instances completes a one-way path for sending OTMA transaction messages asynchronously from a local sending IMS system to a remote receiving IMS system.

Any responses generated by the remote IMS system are queued to a tpipe hold queue in the remote IMS system for asynchronous retrieval.

To define an IMS-to-IMS TCP/IP communications path for OTMA, you must code the following items in the IMS Connect and IMS instances at each side of the connection:

- In the sending IMS Connect, the IMS Connect configuration statements, including the RMTIMSCON statement, which is required for IMS-to-IMS TCP/IP communications, and the DATASTORE statement, which is required for communication between IMS Connect and OTMA.
- In IMS, either the OTMA destination descriptor in the DFSYDTx member of the IMS.PROCLIB data set or the OTMA User Data Formatting exit routine (DFSYDRU0).

Security for the TCP/IP connection can be implemented by using the optional RACF PassTicket support. Transaction authorization can also be implemented in the IMS system.

OTMA requires the RMTIMSCON statement in only the IMS Connect instance that sends messages on the TCP/IP connection.

OTMA sends transaction messages across IMS TCP/IP connections using commit mode 0 and the send-only with acknowledgment protocol. Any responses generated at the remote IMS system to the OTMA transaction messages are queued to a tpipe hold queue for asynchronous retrieval.

You can define a separate TCP/IP connection to return responses back to their originating IMS system; however, the responses must be sent back to the originating IMS system as new transactions for the originating system and the correlation of responses to their original transaction must be managed by your installation.

In IMS Connect, you can monitor, stop, and restart the TCP/IP connections by using IMS Connect WTOR, z/OS MODIFY, or IMS type-2 commands.

Related concepts

“OTMA support for IMS-to-IMS communications” on page 778
You can send OTMA messages from a local IMS system to a remote IMS system by using IMS-to-IMS TCP/IP communications.

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

IMS Connect, OTMA super member, and IMS-to-IMS TCP/IP connections

IMS Connect supports the OTMA super member function with IMS-to-IMS TCP/IP communications. The OTMA super member function can help ensure availability and distribute workloads across IMS Connect instances.

For IMS-to-IMS TCP/IP communications, you can define up to eight connections from IMS Connect to an OTMA super member group. The connections are defined by the IMS Connect DATASTORE configuration

statement. OTMA counts each connection as a target member (tmember). The connections can be from one IMS Connect instance or multiple IMS Connect instances.

If more than eight connections are defined to use the super member group, OTMA uses only the first eight connections. If IMS is restarted when more than eight connections are defined from actively running IMS Connect instances, the same eight connections might not be accepted into the super member group because the connections are joined randomly.

When a connection shuts down, either because it is stopped or IMS Connect is shut down, OTMA removes it from the super member group. If a connection is started, OTMA joins it to the super member group only if the group is not already full.

If a connection is removed from a full super member group, OTMA does not automatically add other connections that tried to join previously. To add a connection for an IMS Connect instance that is already running, either restart the connection or the IMS Connect instance.

Related concepts

"Super member support for IMS-to-IMS communications" on page 779
You can use the OTMA super member function to distribute messages sent to a remote IMS system across multiple local instances of IMS Connect.

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

Related reference

DATASTORE statement (System Definition)

Automatic reconnection attempts for OTMA connections

If IMS Connect cannot establish a connection with a remote IMS Connect instance, IMS Connect automatically attempts to reconnect to the remote IMS Connect instance every 2 minutes.

When AUTOCONN=Y is specified in the RMTIMSCON statement, IMS Connect creates connections to a remote IMS Connect instance at startup. The number of connections that IMS Connect creates at startup is determined by the RESVSOC keyword of the RMTIMSCON configuration statement.

If AUTOCONN=N, IMS Connect creates a connection only when a message is received from OTMA for delivery to the remote IMS system.

In either case, if IMS Connect cannot create the connections due to any of the following reasons, IMS Connect attempts to reconnect to the remote IMS Connect instance every 2 minutes:

- Local TCP/IP is not available
- Remote TCP/IP is not available
- Remote IMS Connect is not available

If any messages are received from OTMA either before, during, or between reconnection attempts, IMS Connect sends a NAK to OTMA that directs OTMA to leave the message at the front of the tpipe queue. When the connection to the remote IMS Connect is established, IMS Connect notifies OTMA to resume sending messages on the connection.

While IMS Connect is attempting to reconnect, the status of the connection is RETRY CONN. Also, the NUMSOC value that indicates how many sockets are open fluctuates. IMS Connect opens a socket when actively trying to reconnect. IMS Connect closes the socket during the 2-minute interval between reconnection attempts.

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

Overview of IMS Connect XML Conversion Support

For certain IMS Connect clients, IMS Connect can convert the XML data contained in an input message into the data structures used by IMS application programs written in either COBOL or PL/I. The data in

the corresponding output message is also converted from programming language of the IMS application program back to the XML data that IMS Connect client expects.

The IMS Connect XML conversion support enables IMS to accept messages in an XML format without having to create or modify IMS application programs to support XML.

The IMS Connect XML conversion support uses the HWSSOAP1 user message exit to identify the appropriate XML adapter and XML converter. The HWSSOAP1 user message exit is used by both IMS Enterprise Suite SOAP Gateway and IMS Web 2.0 Solution for IBM Mashup Center.

IMS Connect calls the XML adapter, which serves as the interface to the XML converter. The XML converter is generated based on either the COBOL copybook or the PL/I source of the IMS application program, depending on which language the IMS application program is written in. After the XML data has been converted into the data structures of the programming language of the IMS application program, the input message is passed to OTMA.

Related concepts

"IMS Connect XML message conversion" on page 209
For some IMS Connect client application programs, IMS Connect can convert XML messages into either COBOL or PL/I, so that you do not need to modify existing IMS application programs to process messages submitted to IMS in XML.

Related tasks

Configuring XML conversion support for IMS Connect clients (System Definition)

IMS Connect support for z/OS Sysplex Distributor

IMS Connect includes a variety of features that facilitate its execution in a z/OS Sysplex Distributor environment.

In a z/OS Sysplex Distributor environment, incoming messages are typically distributed among multiple instances of IMS Connect to balance the workload and increase availability. In such an environment, client applications have no control over which instance of IMS Connect receives their input messages and which IMS Connect receives subsequent requests for asynchronous output. IMS Connect, with OTMA, provides several features to support operating in such an environment, such as rerouting asynchronous output to an alternate tpipe, sharing asynchronous output by using an OTMA super member tpipe, retrieving output from an alternate tpipe queue associated with another client, and purging undeliverable output.

IMS Connect automatically sends a server health status report to z/OS Workload Manager (WLM) when the server is started. The health status is a number in the range 0 - 100 that indicates the health of IMS Connect and is initially set to 100. The health value is defined based on the available socket percentage, which is the number of available sockets as a percentage of the maximum allowable number of sockets that is set by the MAXSOC parameter in the TCPIP configuration statement. The available socket percentage and the corresponding health status number are mapped in the following way:

- If the maximum allowable socket number is equal to or greater than 1000 (MAXSOC >= 1000):
  - If the available socket percentage is 11% - 100%, the health status number is 100.
  - If the available socket percentage is 0% - 10%, the health status number is equivalent to the number of the available socket percentage, which is 0 - 10.
- If the maximum allowable socket number is less than 1000 (MAXSOC < 1000):
  - If the available socket percentage is 21% - 100%, the health status number is 100.
  - If the available socket percentage is 0% - 20%, the health status number is equivalent to the number of the available socket percentage, which is 0 - 20.

An updated health status report is sent when the server health changes.

z/OS Sysplex Distributor uses this information to route incoming connections when the distribution method SERVERWLM is specified on the **VIPADISTRIBUTE** statement of the z/OS TCP/IP profile. You can also set the SHAREPORTWLM parameter for the **PORT** definition to configure Sysplex Distributor to use

the WLM health status to balance incoming connections between two or more instances of IMS Connect that are running on the same host with a shared port.

Related concepts

"Rerouting commit-then-send output" on page 293
You can configure IMS to reroute commit-then-send (commit mode 0) IOPCB output to an alternate OTMA tpipe hold queue for retrieval.

Related tasks

"Sharing asynchronous commit-then-send output: the OTMA super member function" on page 847
Hold-queue-capable OTMA clients, such as IMS Connect, can share asynchronous commit-then-send (CM0) output messages by enabling the OTMA super member function. The OTMA super member function is specifically designed to support multiple instances of IMS Connect in a z/OS Sysplex Distributor environment.

"Purging undeliverable commit-then-send output" on page 291
You can configure OTMA to purge commit-then-send (commit mode 0) IOPCB output when the output cannot be returned to the OTMA client application that initiated the transaction.

"Retrieving output from alternate OTMA tpipe hold queues" on page 335
Client applications can retrieve the asynchronous output or callout messages from an alternate tpipe hold queue by specifying the name of the alternate tpipe as an alternate client ID a RESUME TPIPE call.

Related reference

TCPIP statement (System Definition)

Overview of IMS Connect security

IMS Connect provides different security options depending on whether a client is accessing IMS DB or IMS TM.

IMS DB clients can implement security by using the IMS Connect DB Security user exit routine (HWSAUTH0), a security product such as RACF, or both. For IMS DB clients, IMS Connect also provides support for RACF PassTickets. For Secure Sockets Layer (SSL) support, IMS DB clients can use IBM z/OS Communications Server Application Transparent Transport Layer Security feature (AT-TLS). IMS Connect does not provide SSL support for IMS DB clients.

IMS TM clients can implement security using any combination of the IMS Connect user message exit routines, a user security exit routine, and a security product such as RACF. For IMS TM clients, IMS Connect provides direct support for SSL and support for RACF PassTickets.

For IMS-to-IMS TCP/IP connections, IMS Connect provides optional connection security by using RACF PassTickets.

In an IMS Connect configuration, security can be implemented by using various combinations of the following components:

- On the client side:
  - The client application
  - The server of the client application
- IMS Connect
- A security product, such as RACF
- For IMS TM connections:
  - The IMS Connect user message exit routines
  - OTMA, including the OTMA Resume Tpipe Security user exit (OTMARTUX)
- For IMS DB connections, the IMS Connect DB Security user exit routine
- IMS
- An IMS exit routine

- An IMS application program

When you configure IMS Connect, you can enter your security specifications in the following places:

- HWSCFGxx configuration member
- The RACF FACILITY class

Related concepts

"IMS Connect security support" on page 175
IMS Connect includes a variety of options for implementing and modifying the security checking performed on messages as they arrive in IMS Connect and, for IMS TM connections, as they arrive at the data store.

Related tasks

"RACF PassTicket for IMS Connect client connections to IMS DB" on page 194
You can use RACF PassTickets to authenticate IMS Connect client connections to IMS DB. PassTickets are an alternative to RACF passwords and password phrases and provide better security because PassTickets remove the need to send passwords and password phrases across the network in clear text.

Overview of defining and invoking IMS Connect

The steps below provide a high-level overview of defining and invoking IMS connect.

Configuring IMS Connect includes the following high-level steps.

1. Authorize SDFSRESL to the Authorized Program Facility (APF) by online command or by running a JCL job.
2. Ensure that the z/OS Program Properties Table (PPT) is updated to allow IMS Connect to run in authorized supervisor state and in key 7. The specification in the z/OS PPT must match the program specification in the EXEC statement of the IMS Connect startup JCL. You can specify either BPEINI00 or HWSHWS00.
3. Create an IMS Connect configuration member, and define the IMS Connect configuration statements that IMS Connect uses during initialization.
4. Create the BPE configuration member for IMS Connect.
5. Create the HWSUINIT initialization exit routine, which is not shipped as a load module.
6. If you are connecting to IMS TM, create the user message exit routines that you need, such as HWSSMPL0, HWSSMPL1, or HWSJAVA0. These user message exit routines are also not shipped as load modules.
7. Define IMS Connect security and, optionally, OTMA client security.
8. Optionally, enable the IMS Connect XML message conversion support.

Invoke IMS Connect using either a z/OS procedure or a z/OS job. If you start multiple instances of IMS Connect with the same configuration, a connection outage can occur.

Related concepts

IMS Connect definition and tailoring (System Definition)

Chapter 11. Overview of IMS Connect exit routines

IMS provides a variety of exit routines to support IMS Connect.

IMS Connect exit routines fall into two general categories:

- User message exit routines that manage the messages to and from the various types of IMS Connect TCP/IP clients
- Exit routines that provide general functionality, such as security and routing

The IMS Connect user message exit routines, which are used only with clients connecting to IMS TM, include:

- HWSSMPL0 and HWSSMPL1 user message exits, for user-written IMS Connect client applications
- HWSJAVA0 user message exit, for the IMS TM Resource Adapter
- HWSSOAP1 user message exit, for the IMS Enterprise Suite SOAP Gateway
- HWSCSLO0 and HWSCSLO1 user message exits, for the OM command clients

All of the IMS Connect client user message exits allow you to call the z/OS TCP/IP IMS Listener security exit routine (IMSLSECX), issue the RACF function in these user message exit routines, or use the IMS Connect user RACF function.

Attention: Do not issue any z/OS calls in the user message exit that result in an `MVS WAIT`. If you modify the user message exit and add code that results in an `MVS WAIT`, all work on the TCP/IP PORT will halt until the WAIT has been posted. The user message exits cannot be modified to free any storage passed to the exit, and IMS Connect will not free any storage obtained by the user message exit when the exit returns to IMS Connect. All storage obtained by IMS Connect must be released by IMS Connect and cannot be freed by the User Message Exit without causing failures.

The exit routines that provide general functionality to IMS Connect include:

- The sample IMS Connect Destination Resolution exit routine (HWSYDRU0), which is a modified version of the OTMA Destination Resolution exit routine (DFSYDRU0)
- IMS Connect User Initialization exit routine (HWSUINIT)
- z/OS TCP/IP IMS Listener security exit routine (IMSLSECX)
- IMS Connect Password Change exit routine (HWSPWCH0)
- IMS Connect Event Recorder exit routine (HWSTECL0)
- IMS Connect Port Message Edit exit routine for TCP/IP clients
- IMS Connect DB Routing user exit routine (HWSROUT0)
- IMS Connect DB Security user exit routine (HWSAUTH0)

IMS Connect always loads HWSUINIT and HWSJAVA0, but HWSSMPL0 and HWSSMPL1 are optional and are loaded only if you include them with the TCPIP statement in the IMS Connect member of the IMS.PROCLIB data set (HWSCFGxx). These four exit routines are provided as load modules for ease of use. Source code is also provided so that you can modify the exit routines for your installation.

This topic contains Product-sensitive Programming Interface information.

Related reference

IMS Connect exit routines (Exit Routines)

© Copyright IBM Corp. 1974, 2020

Overview of user message exit routines

For most types of IMS Connect clients, IMS Connect requires the use of a user message exit routine to manage the messages that are received from and sent to the client.

The user message exit routines can perform a number of tasks related to the management of messages, including:

- Translating input messages into the protocol or format required by IMS and the IMS Open Transaction Manager Access (OTMA) component
- Rerouting messages
- Checking security for input messages
- Returning user-defined messages in response to certain user-defined criteria

The following IMS Connect clients are listed with the IMS Connect user message exit that they require:

User-provided clients that access IMS TM
The HWSSMPL0 or HWSSMPL1 user message exit routine, or a user-written user message exit routine.

Optionally, you can also use an IMS Connect Port Message Edit exit routine, which receives control between IMS Connect and the z/OS TCP/IP stack, to modify the format of input and output messages.

The HWSSMPL0 and HWSSMPL1 exit routines and their related macros are shipped with IMS both as source code and as load modules.

IMS TM Resource Adapter
The HWSJAVA0 user message exit routine.

Optionally, you can also use an IMS Connect Port Message Edit exit routine, which receives control between IMS Connect and the z/OS TCP/IP stack, to modify the format of input and output messages.

The HWSJAVA0 exit routine and its related macros are provided as both load modules and source code.

IMS Enterprise Suite SOAP Gateway
The HWSSOAP1 user message exit routine, which is provided as object code only.

Optionally, you can also use an IMS Connect Port Message Edit exit routine, which receives control between IMS Connect and the z/OS TCP/IP stack, to modify the format of input and output messages.

IBM WebSphere DataPower®
The HWSDPWR1 message exit routine, which is provided as object code only.

Optionally, you can also use an IMS Connect Port Message Edit exit routine, which receives control between IMS Connect and the z/OS TCP/IP stack, to modify the format of input and output messages.

Clients that access IMS DB, such as the IMS Universal drivers.
IMS Connect does not support user message exit routines for clients that access IMS DB, such as the IMS Universal drivers and user-provided clients that use the Distributed Relational Database Architecture (DRDA) interface. Instead, you can use the following IMS Connect exit routines for message routing, security, and message editing:

- IMS Connect DB Routing user exit routine (HWSROUT0)
- IMS Connect DB Security user exit routine (HWSAUTH0)
- IMS Connect Port Message Edit exit routine

Clients that submit commands to the Operations Manager (OM)
The HWSCSLO0 or HWSCSLO1 user message exit routines, which are provided as object code only.

IMS-to-IMS TCP/IP connections do not use an IMS Connect user message exit routines.

Related reference
IMS Connect user message exit routines (Exit Routines)

Security and the IMS Connect user message exit routines

IMS Connect user message exit routines can perform security checking. If configure your exit routines to check security, you must provide a security exit or use the z/OS TCP/IP IMS Listener security exit (IMSLSECX).

IMS does not provide a sample security exit due to the many options available for security and the fact that most installations have their own specific security method. The call to RACF is performed by IMS Connect if RACF parameters are provided in the OTMA header when the message exit returns the message.

IMSLSECX is the name of the security exit called by the following IMS Connect user message exit routines:

- HWSSMPL0
- HWSSMPL1
- HWSSOAP1
- HWSCSLO0

If you use HWSSMPL0 or HWSSMPL1, you can change the name of the security exit that they call by changing EXTRN IMSLSECX to a name of your choice. If you change the name of the security exit, you must define the security exit in the HWSSMPL0 or HWSSMPL1 user message exit.

You can also provide the name of the security exit called by HWSJAVA0 and define it in the HWSJAVA0 message exit.

For more information about the IMSLSECX exit routine, see:

- *IMS Version 15 Exit Routines*
- *z/OS Communications Server: IP IMS Sockets Guide*

Related concepts

"IMS Connect security exit routine" on page 183
If any IMS Connect user message exit routine performs security checking, you must provide a security exit routine or use the z/OS TCP/IP IMS Listener security exit routine (IMSLSECX).

"IMS Connect security and the OTMARTUX user exit" on page 183
The OTMA Resume TPIPE Security user exit (OTMARTUX) is not an IMS Connect exit routine, but it is one of two possible methods that you can use to secure messages queued on the OTMA asynchronous hold queue. The other method is to use an external security product, such as RACF. You can use the OTMARTUX user exit and an external security product each by itself or in combination.

Related reference

"HWSSMPL0 and HWSSMPL1 security actions" on page 183
The sample user message exits HWSSMPL0 and HWSSMPL1 always perform certain security actions and perform other security actions only when the IMSLSECX security exit is or is not called.

User-defined messages

The IMS Connect HWSJAVA0, HWSSMPL0, and HWSSMPL1 user message exits can return user-defined messages to the IMS Connect client when criteria that you define are met.

Any client application programs that might receive a user-defined message must be able to recognize and process user-defined messages and any associated return and reason codes. When a user-defined message is returned to the client application program, the original input message is not sent to IMS.

When a user message exit returns a user-defined message, the original input message is not passed on

Upon returning one of these user-defined messages, you can also have the user message exit request that IMS Connect either keep the socket connection open or close it, depending on your needs.

User-defined messages can be 1 to 128 characters in length. Any message longer than 128 characters is truncated.

If the message you define is longer than the client input message that is received by the user message exit routine from the IMS Connect, increase the buffer size used by the exit routine by specifying the needed extra bytes in the EXPINI_BUFINC field returned by the exit.

To request that IMS Connect keep a socket connection open after returning a user defined message, the exit routine must set EXPREA_RETCODE to 20 (X'14'). To terminate a socket connection, the exit routine must set EXPREA_RETCODE to 4 (X'04'). Because other factors can cause IMS Connect to terminate a connection, specifying 20 in EXPREA_RETCODE does not guarantee that the socket connection will remain open.

For HWSJAVA0, you can return user-defined message text or only a return code and reason code. To return user-defined message text, the HWSJAVA0 user message exit must set OMUSER_RETCODE to 48 (X'30') and OMUSR_RSNCODE to ICONSUCC. Any other combination of values for OMUSER_RETCODE and OMUSER_RSNCODE returns a return and reason code without user-defined message text.

For HWSSMPL0 and HWSSMPL1, you return user-defined message text by placing the message text in the output message buffer.

Related reference

User message exit routines HWSSMPL0 and HWSSMPL1 (Exit Routines)

IMS TM Resource Adapter user message exit routine (HWSJAVA0) (Exit Routines)

Overview of function-specific exit routines

IMS provides several IMS Connect exit routines that perform specific functions for IMS Connect for increased flexibility.

You can use the following function-specific exit routines with IMS Connect:

- IMS Connect User Initialization exit routine (HWSUINIT)
- IMS Connect DB Routing user exit routine (HWSROUT0)
- IMS Connect DB Security user exit routine (HWSAUTH0)
- IMS Connect sample OTMA Destination Resolution exit routine (HWSYDRU0)
- z/OS TCP/IP IMS Listener security exit (IMSLSECX)
- IMS Connect Event Recorder exit routine (HWSTECL0)
- IMS Connect Password Change exit routine (HWSPWCH0)

Related reference

IMS Connect function-specific exit routines (Exit Routines)

Macros that support IMS Connect exit routines

IMS provides macros that support the IMS Connect exit routines.

Macros used for IMS Connect Exit Routines

The macros include:

HWSAUTPM
Maps the parameter list for the IMS Connect DB Security user exit routine (HWSAUTH0). A copy of this macro is in SDFSMAC.

HWSEXPIO
Maps the parameter list for the IMS Connect Port Message Edit exit routine (HWSPIOX0). A copy of this macro is in SDFSMAC.

HWSEXPRM
Maps the parameter list that is passed to the user exit routine on each subroutine call. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSOMPFX
Maps the OTMA message prefix format to the output buffer that the user exit routine returns on each READ subroutine call and the input buffer that is passed to the user exit on each XMIT subroutine call. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSIMSCB
Maps the IMS request message (IRM) header and BPE header formats used by the HWSSMPL0 and HWSSMPL1 user message exit routines. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSIMSEA
Maps the storage area used by the HWSSMPL0 and HWSSMPL1 user message exit routines. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSROUPM
Maps the parameter list that is passed to the IMS Connect DB Routing user exit routine (HWSROUT0) on each subroutine call. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSXIB
Maps the exit interface block used by IMS Connect user message exit routines and the HWSUINIT exit routine. Contains the addresses of the data store list (HWSXIBDS) and the HWSXIB1 control block used by the IMS Connect DB Routing user exit routine. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSXIB1
Maps the exit interface block used by the HWSROUT0 user exit routine. HWSXIB1 contains the address of the ODBM list and optional user data. The HWSXIB1 exit interface block is pointed to by HWSXIB. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSXIBDS
Maps the entry in the exit interface block data store list used by the IMS Connect user message exit routines and the HWSUINIT exit routine. The list contains the data store name, the data store availability and status information, and a user field. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro.

HWSXIBOD
Maps the ODBM list that contains the name and status of each ODBM instance known to IMS Connect, as well as a user field and the names and statuses of the IMS aliases associated with each ODBM instances. The address of HWSXIBOD is stored in the HWSXIB1 exit interface block. A copy of this macro is in SDFSMAC. To see the structure, assemble the macro or refer to the macro prologue.

Exit interface blocks

IMS Connect provides exit interface blocks to support the processing of IMS Connect exit routines that support connections to either IMS DB or IMS TM systems.

XIB exit interface block for connections to IMS TM

IMS Connect provides the XIB exit interface block to support the processing of IMS Connect user message exit routines that are used when connecting to IMS TM. You can use the XIB exit interface block and the user area it includes to store information that is used by your exit routines.

IMS Connect stores the following types of information in the exit interface block:

- Information about the XIBDS.
- Information about the IRM architecture level used.
- User data.

You can also use the XIB_USERAREA field of an exit interface block for any purpose. You can code the IMS Connect User Initialization exit routine (HWSUINIT) or your IMS Connect user message exit routine to use the XIB_USERAREA field.

The exit interface block is mapped by the HWSXIB macro.

Format of the XIB exit interface block

The XIB exit interface block is mapped by the HWSXIB macro.

The XIB exit interface block is provided to support the processing of IMS Connect exit routines. For example, the XIB exit interface block can be used to store the address of a table that an IMS Connect user message exit routine uses during processing.

The following table describes the fields and field offsets of the exit interface block.

Field	Dec Offset	Hex Offset	Length	Value
XIB_HEADER	0	X'00'	0	
XIB_EYE	0	X'00'	4	Eye catcher.
XIB_DATASTORES	4	X'04'	4	The address of the XIBDS data store list.
XIB_UFLD_CNT	8	X'08'	4	The number, in hexadecimal format, of fullword user fields in the XIB_USERAREA.
XIB_XIBDS_LEN	12	X'0C'	2	Length of the XIBDS entries.
XIB_ARCHLVL	14	X'0E'	1	Architecture level of the XIB exit interface block. **X'01'** XIB_ARCH1 - Architecture level 1 **X'02'** XIB_ARCH2 - Architecture level 2 **X'03'** XIB_ARCH3 - Architecture level 3 **XIB_ARCH3** Highest architecture level
	15	X'0F'	1	Reserved.
XIB_VERSION	16	X'10'	4	The IMS version of the IMS Connect instance.
XIB_XIB1	20	X'14'	4	Address of the XIB1 exit interface block for connections to IMS DB.
	24	X'18'	12	Reserved.
XIB_USERAREA	36	X'24'	0F	Start of the user area of the XIB.

XIBDS exit interface block for IMS TM data store information

For connections to IMS TM, IMS Connect keeps track of the status of IMS data stores in entries in the XIBDS exit interface block for data store information. IMS Connect user message exit routines can reference the XIBDS exit interface block to make routing decisions for incoming messages based on the status of the data store.

IMS Connect stores the following types of information about IMS data stores in the exit interface block data store entries:

- Availability of the IMS data stores.
- Whether an IMS data store is running on a different z/OS image than IMS Connect.
- Whether support for cascading global RRS transactions to an IMS data store that is running on a different z/OS image is enabled.

- The state of the IMS data stores; that is, how well the data store is processing messages and, if processing is degraded or completely unavailable, what conditions in the IMS data store might be causing the degraded or unavailable state.
- A time stamp that records the time of the last status change or heartbeat message from OTMA.

IMS Connect updates the state information for an IMS data store when a data store connection is first established and when OTMA notifies IMS Connect of changes in the state of the data store.

OTMA issues a heartbeat message every 60 seconds to indicate that the data store is still communicating. If the time stamp in a data store entry is older than 60 seconds, OTMA could be experiencing problems.

You can also use the XIBDS_USER field of an exit interface block data store entry to for any purpose. You can code the IMS Connect User Initialization exit routine (HWSUINIT) to set the XIBDS_USER field during IMS Connect startup.

The XIBDS exit interface block data store entries are mapped by the HWSXIBDS macro as shown in the following table.

Format of XIBDS exit interface block

The XIBDS exit interface block data store entries are mapped by the HWSXIBDS macro.

The following table describes the fields and field offsets of an entry in the exit interface block.

Field	Dec Offset	Hex Offset	Length	Value
XIBDS_NAME	0	X'00'	8	Name of the data store.
XIBDS_STATUS	8	X'08'	1	Availability of the data store. The possible values are: **X'00'** The data store is not active. **X'01'** The data store is active. **X'02'** The data store is disconnected.

Field	Dec Offset	Hex Offset	Length	Value
XIBDS_FLAG	9	X'09'	1	Data store entry flags. **X'80'** Identifies the last entry in the exit interface block. **X'40'** The IMS data store is running on a different z/OS image (LPAR) than IMS Connect. This flag can be set only when the data store connection is active in IMS Connect and the IMS data store is active in the XCF group. **X'20'** Cascaded transaction support for global RRS transactions synchlevel=2 (syncpoint) has been enabled for the data store. This flag can be set only when the XIBDS_FLAG field also contains X'40', which indicates that IMS Connect and the IMS data store reside on different z/OS images (LPARs). **X'10'** Indicates that the version of the IMS data store is stored at offset 10. **X'08'** Entry for the IMS data store. **X'04'** Entry for the IMSPlex.
IMS version	10	X'0A'	2	The version of the IMS data store is included here only if the X'10' flag is on in the XIBDS_FLAG field at offset 9. The format of the version number is: • IMS Release (1 byte) • IMS Level (1 byte)
XIBDS_USER	12	X'0C'	4	User field
XIBDS_ST_STATUS	16	X'10'	2	The overall state of the IMS data store. The values in this field indicate how well the data store is processing input messages. The possible values are: **3** Normal state: IMS is available and is processing input messages normally. **2** Degraded state: IMS is processing OTMA messages slowly. One or more conditions indicate that IMS is not processing input messages as quickly as it should. **1** Unavailable state: IMS can no longer accept input messages for processing. One or more severe conditions prevent IMS from processing OTMA messages.

Field	Dec Offset	Hex Offset	Length	Value
XIBDS_ST_SVRSTT			0	The beginning of the fields used to identify the conditions that are contributing to the unavailable state of data store processing.
XIBDS_ST_SVRFLG1	18	X'12'	1	Reserved
XIBDS_ST_SVRFLG2	19	X'13'	1	Reserved
XIBDS_ST_SVRFLG3	20	X'14'	1	Reserved
XIBDS_ST_SVRFLG4	21	X'15'	1	**X'01** The data store is flooded with messages from this IMS Connect instance and is no longer accepting input from this instance.
XIBDS_ST_WRNSTT			0	The beginning of the fields used to identify the conditions that are contributing to the degraded state of data store processing.
XIBDS_ST_WRNFLG1	22	X'16'	1	**X'80'** The global number of messages that are waiting to be processed by the data store has passed 80 percent of the maximum allowable number of waiting messages that is defined for all OTMA clients in this z/OS cross-system coupling facility (XCF) group. If the number of waiting messages reaches 100 percent of maximum allowable number, OTMA sets a flood condition and rejects all incoming messages from all OTMA clients in the XCF group.
XIBDS_ST_WRNFLG2	23	X'17'	1	Reserved
XIBDS_ST_WRNFLG3	24	X'18'	1	Reserved
XIBDS_ST_WRNFLG4	25	X'19'	1	**X'01'** The number of messages that are waiting to be processed by the data store has passed 80 percent of the maximum allowable number for waiting messages that is defined in OTMA for this instance of IMS Connect. If the number of waiting messages reaches 100 percent of the maximum allowable number, OTMA rejects all incoming messages from this instance of IMS Connect.
XIBDS_ST_UTC	26	X'1A'	12	The UTC time at which this status was issued by OTMA

XIB1 exit interface block for connections to IMS DB

The XIB1 exit interface block supports IMS Connect exit routines on connections to IMS DB through the Open Database Manager (ODBM).

The XIB1 exit interface block contains the address of the XIBOD exit interface block, which stores information about ODBM and IMS data stores. You can also use XIB1 exit interface to store user data for use by exit routines, such as the IMS DB Routing user exit routine (HWSROUT0).

You can use the XIB1_USERAREA field of an exit interface block for any purpose. Both the IMS Connect User Initialization exit routine (HWSUINIT) and the IMS Connect DB Routing user exit routine can be coded to use the XIB1_USERAREA field.

The exit interface block is mapped by the HWSXIB1 macro.

Format of the XIB1 exit interface block

The XIB1 exit interface block is mapped by the HWSXIB1 macro.

The following table describes the fields and field offsets of the XIB1 exit interface block.

Field	Dec Offset	Hex Offset	Length	Value
XIB1_HEADER	0	X'00'	0	Align on doubleword.
XIB1_EYE	0	X'00'	4	Eye catcher.
XIB1_ODBMS	4	X'04'	4	Address of the XIBOD exit interface block that contains ODBM and data store information.
XIB1_UFLD_CNT	8	X'08'	4	User field count.
XIB1_ARCHLVL	12	X'0C'	1	Architecture level of the XIB1 exit interface block. **X'01'** XIB1_ARCH1 - Architecture level 1 **XIB1_ARCH1** Highest archictecture level
Reserved	13	X'0D'	3	Reserved for IMS Connect.
XIB1_XIB	16	X'10'	4	Address of the XIB.
Reserved	20	X'24'	16	Reserved.
XIB1_USERAREA	36	X'24'	0F	Start of the user area of the XIB1.

XIBOD exit interface block for ODBM and IMS DB data store information

For connections to IMS DB, IMS Connect keeps track of the status of Open Database Manager (ODBM) instances and IMS data stores in entries in the *XIBOD exit interface block for ODBM and data store information*. The IMS Connect DB Routing user exit routines can reference the XIBOD to make routing decisions for incoming messages based on whether the ODBM instances and data stores are active.

For the ODBM instances known to IMS Connect, the XIBOD keeps track of the following ODBM states:

- ODBM is running and connected to IMS Connect.
- ODBM is running but not connected to IMS Connect. You can issue the IMS Connect WTOR command **STARTOD** or the IMS Connect type-2 command **UPDATE IMSCON TYPE(ODBM) START(COMM)** to establish a connection to ODBM.
- ODBM is running but not reachable because the Structured Call Interface (SCI) on the LPAR that the ODBM is running on is down. Restart the SCI to restore the connection between IMS Connect and ODBM.
- ODBM is not running. ODBM must be restarted before a connection can be made between IMS Connect and ODBM.

The XIBOD block also stores ODBM version information.

Data stores are known to IMS Connect by the alias names that are assigned to the data store in the ODBM configuration member CSLDCxxx during ODBM system definition. For IMS Connect to route an incoming request to a specific data store, the alias name of the data store must be active in both the ODBM in which it is defined and in IMS Connect.

For each alias name defined to the ODBM instances that are known to IMS Connect, the XIBOD keeps track of the following states of the connection to the data store represented by the alias name:

- The connection to the data store is complete. The alias name is active in both the ODBM instance and in IMS Connect (XIBOD_ICACTIVE EQU X'20').
- The connection to the data store is not complete. The alias name is active in ODBM, but has never been activated in IMS Connect (XIBOD_IACTIVE EQU X'80').
- The connection to the data store is not complete. The alias name was active in both ODBM and IMS Connect, but is no longer active in IMS Connect. The alias name is still active in ODBM (XIBOD_ICINACTIVE EQU X'10').
- The connection to the data store is not complete. The alias name was active in both ODBM and IMS Connect, but is no longer active in ODBM. Consequently, the alias name is no longer active in IMS Connect (XIBOD_IINACTIVE EQU X'40').

You can also use the XIBOD_USER field of an exit interface block data store entry for any purpose. You can code the IMS Connect User Initialization exit routine (HWSUINIT) to set the XIBOD_USER field during IMS Connect startup.

The exit interface block data store entries are mapped by the HWSXIBOD macro.

Format of XIBOD exit interface block

The XIBOD exit interface block for ODBM and data store information is mapped by the HWSXIBOD macro.

Map of the ODBM entries in the XIBOD exit interface block

The ODBM entries in the XIBOD exit interface block are mapped by the HWSXIBOD DSECT in the HWSXIBOD macro, as shown in the following table.

Field	Dec Offset	Hex Offset	Length	Value
XIBOD_HDR	0	0	0	
XIBOD_EYE	0	0	4	Character data. **XIBOD** XIBOD_EYEID: The eye catcher.
XIBOD_NAME	4	X'04'	8	Name of the Open Database Manager (ODBM) instance.

Field	Dec Offset	Hex Offset	Length	Value
XIBOD_OSTATUS	12	X'0C'	1	The status of ODBM. The possible values are: **X'80'** XIBOD_OACTIVE: ODBM is running and connected to IMS Connect. **X'40'** XIBOD_OINACTIVE: ODBM is running but not connected to IMS Connect. You can issue the IMS Connect WTOR command **STARTOD** or the IMS Connect type-2 command **UPDATE IMSCON TYPE(ODBM) START(COMM)** to establish a connection to ODBM. **X'20'** XIBOD_ODISC: ODBM is not running or is no longer a member of the IMSplex. ODBM must be restarted in the IMSplex before a connection can be made between IMS Connect and ODBM. **X'10'** XIBOD_ONOTRCHB: ODBM is running but not reachable because the non-local Structured Call Interface (SCI) that resides on the same LPAR as the ODBM is down. The ODBM's SCI must be restarted to restore the connection between IMS Connect and ODBM.
XIBOD_ODBMRRS	13	X'0D'	1	The character Y or N. Y indicates that ODBM is using z/OS Resource Recovery Services (RRS). N indicates that ODBM is not using RRS.
Reserved	14	X'0D'	2	Reserved
XIBOD_ODBMVER	16	X'10'	4	ODBM version number
XIBOD_USER	20	X'14'	4	User field
XIBOD_NEXTODBM	24	X'18'	4	Address of the next ODBM
XIBOD_NEXTATBL	28	X'1C'	4	Address of the table of alias names defined to the ODBM at address XIBOD_NEXTODBM
Reserved	32	X'20'	16	Reserved
XIBOD_IMSATABLE	48	X'30'	512	Character data. The alias name table has 32 alias name entries. Each alias name entry is 16 bytes long. See the following table for a map of an alias name entry.

Map of the alias name entries in the XIBOD exit interface block

The alias name entries are mapped by the XIBOD_IMSAENT DSECT in the HWSXIBOD macro, as shown in the following table. The alias name table in the XIBOD exit interface block contains 32 alias name entries.

Field	Dec Offset	Hex Offset	Length	Value
XIBOD_IMSA	0	0	4	Character data. The alias name.

Field	Dec Offset	Hex Offset	Length	Value
XIBOD_ISTATUS	4	X'04'	1	The status of the connection to the data store named by the alias. **X'80'** XIBOD_IACTIVE: The connection to the data store is not complete. The alias name is active in ODBM, but has never been activated in IMS Connect. **X'40'** XIBOD_IINACTIVE: The connection to the data store is not complete. The alias name was active in both ODBM and IMS Connect, but is no longer active in ODBM. Consequently, the alias name is no longer active in IMS Connect. **X'20'** XIBOD_ICACTIVE: The connection to the data store is complete. The alias name is active in both the ODBM instance and in IMS Connect. **X'10'** XIBOD_ICINACTIVE: The connection to the data store is not complete. The alias name was active in both ODBM and IMS Connect, but is no longer active in IMS Connect. The alias name is still active in ODBM. **X'08'** XIBOD_DELETED: The IMS alias name has been deleted from the ODBM CSLDCxxx configuration member and can no longer be used.
Reserved	5	X'05'	3	Reserved for IMS Connect.
Reserved	8	X'08'	8	Reserved for IMS Connect.

Related reference

IMS Connect UPDATE ODBM command (Commands)

STARTOD command (Commands)

UPDATE IMSCON TYPE(ODBM) command (Commands)

Chapter 12. IMS Connect support for IMSplex and shared queues

IMS Connect can send and receive IMS Operations Manager (OM) commands and response string messages between a TCP/IP client and the OM in an IMSplex by using the IMS Structured Call Interface (SCI).

IMS Connect can send and receive IMS Operations Manager (OM) commands and response string messages between a TCP/IP client and the OM in an IMSplex by using the IMS Structured Call Interface (SCI). IBM Management Console for IMS and Db2 for z/OS is an IBM-provided client that uses this interface.

The following subsections provide detailed information about the environment requirements and how to set up IMS Connect to support OM.

IMS Connect support for IMSplex

TheIMS Connect support for IMSplex enables access to an OM in an IMSplex fromIBM Management Console for IMS and Db2 for z/OS (Management Console) or other OM command client through TCP/IP.

The IMSplex support accesses OM through the IMS SCI. The IMSplex statement in the IMS Connect configuration file (HWSCFGxx) defines IMS Connect for IMSplex support. If the IMSplex statement is omitted, then IMSplex support is not available.

IMSplex support sends IMS command string messages directly for a client (for example, the Management Console) to a selected OM within an IMSplex. One or more IMSplexes can be defined to IMS Connect to receive Management Console command messages. SCI is used to communicate between IMS Connect and the IMSplex. To gain access to the selected OM, include the IMSPLEX configuration statement in the IMS Connect configuration member in the IMS.PROCLIB data set.

The same security method (authentication of the userid, groupid, and password; or authentication of the userid, groupid, userid application name, and pass ticket) used for accessing data stores also applies to IMSplex support. An IMS Connect command message exit performs similar functions as an IMS Connect user message exit.

The OM command client exit routines (HWSCSLO0 and HWSCSLO1) are designed specifically for IMSplex support. The HWSCSLO0 and HWSCSLO1 exit routines are similar to the other user message exit routines provided by IMS Connect for client access to the data store. The HWSCSLO0 and HWSCSLO1 message exits are designed to be used only by OM command clients and cannot be used by any client that sends messages to a data store. The HWSCSLO0 and HWSCSLO1 exit routines process only command string messages from OM command clients and are delivered as object code only.

IMSplex support environment

IMS Connect requires that IMS, the CSL Operations Manager (OM), and the CSL Structured Call Interface (SCI) be running to communicate with OM.

IMS must be running in the same z/OS image as IMS Connect or a different z/OS image on the same IMSplex. The OM must be in the same z/OS image or a different z/OS image in the same IMSplex. The SCI must be in the same z/OS image as IMS Connect.

IMS Connect can be brought up before or after IMS, SCI, OM, and RM. During IMS Connect initialization, connection to SCI is made. IMS Connect attempts to connect to SCI for 30 minutes. If SCI connection is not made, then an **OPENIP** command will need to be issued to connect to the SCI after the SCI has been initialized. If the SCI terminates normally or abnormally, IMS Connect will automatically reconnect to the SCI when the SCI is restarted.

© Copyright IBM Corp. 1974, 2020

Related concepts

IMS system administration considerations and tasks (System Administration)

Installing IMS Connect support for IMSplex

The IMS Connect support for an IMSplex requires the CSL Operations Manager (OM), the CSL Structured Call Interface (SCI), and TCP/IP.

IMS must be installed to use the IMS Connect IMSplex support. Ensure that the IMS.SDFSRESL data set has been added to the STEPLIB to enable access to SCI for IMS Connect IMSplex support.

OM requires the appropriate IMSplex component (IMS, IMS Connect, RM, or ODBM) to be running to provide the OM command capability. IMS Connect IMSplex support does not require these components to be running. However, if these components are not running, none of the commands can be processed.

Install or modify the following components of the IMS Connect support for IMSplex in the following order:

1. The IMS Connect configuration file:
   a. Add the IMSplex statement.
   b. Add the HWSCSLO0 exit to the TCPIP statement EXIT= parameter.
2. The BPE configuration file:
   a. Add OMDR and HWSO statements, if the IMS Connect trace entries are listed separately.
3. The OM command client:
   a. Identify the IMS Connect HWS ID= value.
   b. Identify the IMS Connect IMSPLEX tmember= value.

When installation is complete, start the following features in order:

1. Start SCI.
2. Start OM.
3. If you are using RM, start RM.
4. Start IMS.

You can start IMS Connect before, during, or after the steps listed above. IMS Connect attempts to connect to SCI for 30 minutes. If SCI is not brought up or the connection attempt fails, you must issue an `OPENIP` *`imsplex_name`* command.

The following IMS Connect commands are enabled by IMSplex support:

- `STOPIP`
- `OPENIP`
- `VIEWIP`

Retrieving ALTPCB output in a shared queues environment

In a shared queues environment, to retrieve messages inserted to an ALTPCB output queue by IMS systems to which IMS Connect is not directly connected, use the OTMA super member function.

To use the OTMA super member in a configuration that includes multiple IMS systems, you must have shared queues enabled for all those IMS systems. If a back-end IMS system creates ALT-PCB output in the super member-enabled environment, the output can be retrieved from any front-end IMS with an OTMA client in the super member set.

To activate the super member function:

Specify a 1- to 4-character super member name in the SMEMBER parameter on either the HWS statement or the DATASTORE statement in the IMS Connect configuration member (HWSCFGxx).

Super member names must be unique and cannot be the same as existing OTMA member names.

Chapter 13. IMS Connect security support

IMS Connect includes a variety of options for implementing and modifying the security checking performed on messages as they arrive in IMS Connect and, for IMS TM connections, as they arrive at the data store.

For connections to IMS TM, IMS Connect provides two options for checking security within IMS Connect: you can configure IMS Connect to call RACF directly or you can have the IMS Connect user message exit routines call a Security user exit routine.

For connections to IMS DB, the IMS Connect DB Security user exit routine (HWSAUTH0) authenticates the user and you can use RACF as well. For connections to IMS DB, IMS Connect does not check the authority of the user to perform any action, but can pass a RACO token to the CSL Open Database Manager for the purposes of authorization.

For connections to IMS Operations Manager (OM) for command requests, you can configure IMS Connect to call RACF and use RACF PassTickets for user authentication.

For connections to remote instances of IMS Connect that support IMS-to-IMS TCP/IP communications, IMS Connect supports RACF PassTickets and the establishment of trusted user connection status.

Additional security features provided by IMS Connect include:

- Password management support
- Trusted-user classification for messages arriving at the data store
- OTMA accessor environment element (ACEE) timeout specification support
- For connections from clients connecting to IMS DB:
  - Secure Sockets Layer (SSL) support. To use SSL to secure connections from clients connecting to IMS DB, you can use IBM z/OS Communications Server Application Transparent Transport Layer Security feature (AT-TLS).
  - Support for RACF PassTickets.
- For IMS TM clients, IMS Connect provides support for RACF PassTickets.
- Support for passing to and from IMS the security credentials, including the network user ID and network session ID, that are entered by a user in a distributed environment.

IMS Connect support for RACF

IMS Connect can be configured to call RACF directly and to support RACF PassTickets.

By default, IMS Connect does not call RACF. When IMS Connect is configured to call RACF, IMS Connect can validate the user IDs and passwords on incoming messages with RACF directly.

When configured for direct RACF support, IMS Connect also supports RACF PassTickets.

If RACF is configured to support mixed-case passwords, you can also configure IMS Connect to support mixed-case password support in IMS Connect.

IMS Connect calls RACF by issuing the RACF command **RACROUTE REQUEST=VERIFY** to verify the user IDs and passwords received from clients in the IRM of incoming messages. You can also define a default RACF ID for IMS.

If a RACF security failure occurs, IMS Connect includes the return code from the **RACROUTE REQUEST=VERIFY** command in the request status message (RSM) for diagnostic purposes.

If RACF is used to verify sign-ons from IMS Connect clients and the user ID or password provided is invalid, you can enable a generic return code or message to be returned by IMS Connect instead of the actual RACF or IMS return code. By enabling a generic return code or message to be returned, you can inhibit access to information about RACF-verified sign-ons until valid user IDs and passwords are provided.

© Copyright IBM Corp. 1974, 2020

If you configure IMS Connect to call RACF, evaluate the impact of the RACF calls on IMS Connect performance. Consider enabling the RACF user ID cache to improve performance.

Related tasks

"IMS Connect password management" on page 190
IMS Connect provides several features to help you manage RACF passwords. Some of these features only apply when IMS Connect is configured to call RACF directly.

Related reference

"HWSSMPL0 and HWSSMPL1 security actions" on page 183
The sample user message exits HWSSMPL0 and HWSSMPL1 always perform certain security actions and perform other security actions only when the IMSLSECX security exit is or is not called.

Enabling generic return codes or message for RACF verifications

If RACF is used to verify sign-ons from IMS Connect clients, you can enable a generic return code or message to be returned by IMS Connect instead of the actual RACF or IMS return code if the user ID or password provided is invalid. By enabling a generic return code or message to be returned, you can inhibit access to information about RACF-verified sign-ons until valid user IDs and passwords are provided.

Before you enable a generic return code or message to be returned by IMS Connect instead of the actual RACF or IMS return code, ensure that the IRRSPW00 module, which is provided by RACF, is included in one of the following locations:

- LPA
- A library in LINKLIST
- A library in LINKLIST

To enable a generic return code or message to be returned by IMS Connect instead of the actual RACF or IMS return code, use one of the following methods:

- In the HWS statement of the HWSCFG*xx* member of the IMS PROCLIB data set, specify **RACFGENRC=Y**.
- Issue the **UPDATE IMSCON TYPE(CONFIG)** command with **SET(RACFGENRC(ON))** specified.

Related reference

HWS statement (System Definition)
QUERY IMSCON TYPE(CONFIG) command (Commands)
UPDATE IMSCON TYPE(CONFIG) command (Commands)

Enabling RACF security checking in IMS Connect

You can enable RACF security checking in IMS Connect either by specifying RACF=Y in the HWS configuration statement or by issuing an online IMS Connect command.

To enable RACF in the HWS configuration statement, add the RACF=Y parameter. For example:

```
HWS ID=HWS01 RACF=Y
```

To enable RACF online:

- Issue any one of the following IMS Connect commands:
  - The IMS type-2 format command, **UPDATE IMSCON TYPE(CONFIG) SET(RACF(ON))**
  - The WTOR format command, **SETRACF ON**
  - The z/OS MODIFY command format, **UPDATE MEMBER TYPE(IMSCON) SET(RACF(ON))**
- Verify that RACF security checking is enabled by issuing any of the following IMS Connect commands:
  - The IMS type-2 format command, **QUERY IMSCON TYPE(CONFIG) SHOW(RACF)**
  - The WTOR format command, **VIEWHWS**
  - The z/OS MODIFY command format, **QUERY MEMBER TYPE(IMSCON) SHOW(ALL)**

When RACF is enabled, the output from the IMS type-2 format command **QUERY IMSCON TYPE(CONFIG) SHOW(RACF)** shows a Y in the "Racf" column of the command output.

The output from both the WTOR and z/OS MODIFY commands includes a line similar to the following: `HWSC0001 HWSID=HW01 RACF=Y`.

Related reference

IMS Connect UPDATE MEMBER command (Commands)
SETRACF command (Commands)
UPDATE IMSCON TYPE(CONFIG) command (Commands)
QUERY IMSCON TYPE(CONFIG) command (Commands)
HWS statement (System Definition)
IMS Connect QUERY MEMBER command (Commands)
VIEWHWS command (Commands)

Enabling RACF security statistics for IMS Connect

If IMS Connect is configured to call RACF, you can enable RACF security statistics to be recorded and updated when IMS Connect issues the RACF call **RACROUTE REQUEST=VERIFY**. You can enable RACF statistics to be recorded and updated for ODBM client connections to IMS DB and for OTMA client connections to IMS TM. After you enable RACF statistics, the statistics are updated no more than once per day.

Before you enable RACF security statistics to be recorded when IMS Connect issues the RACF call **RACROUTE REQUEST=VERIFY**, ensure that RACF=Y is specified in the HWS statement of the HWSCFG*xx* IMS PROCLIB member. If RACF=N is used in the HWS statement, IMS Connect does not issue the RACF call **RACROUTE REQUEST=VERIFY**; therefore, RACF statistics will not be recorded.

To enable RACF statistics for ODBM client connections to IMS DB, use one of the following methods:

- To enable RACF statistics in the ODACCESS statement of the HWSCFG*xx* member of the IMS PROCLIB data set, specify ODRACFST=Y.
- To enable RACF statistics by updating the RACF statistics option online, use the ODRACFST(ON) keyword on the **UPDATE IMSCON TYPE(CONFIG)** command.

To enable RACF statistics for OTMA client connections to IMS TM, use one of the following methods:

- To enable RACF statistics in the HWS statement of the HWSCFG*xx* member of the IMS PROCLIB data set, specify TMRACFST=Y.
- To enable RACF statistics by updating the RACF statistics option online, use the TMRACFST(ON) keyword on the **UPDATE IMSCON TYPE(CONFIG)** command.

After you enable RACF statistics, IMS Connect uses the STAT=ASIS parameter on the **RACROUTE REQUEST=VERIFY** call. With STAT=ASIS, the RACF messages and statistics are controlled by the installation's current options on the RACF **SETROPTS** command.

After you enable RACF statistics, the statistics are recorded by RACF no more than once per day to a system management facility (SMF) data set or log stream. The SMF data set or log stream that is used to record the RACF statistics is specified in the RACF configuration.

Related reference

ODACCESS statement (System Definition)
QUERY IMSCON TYPE(CONFIG) command (Commands)
UPDATE IMSCON TYPE(CONFIG) command (Commands)

IMS Connect default RACF user ID

You can set a default RACF user ID for IMS Connect to use when the IRM of an input message either does not contain the IRM_RACF_USERID field or the IRM_RACF_USERID field is blank.

When the default RACF user ID is used, IMS Connect passes it in the OMSECUID field of the input message to OTMA. When OTMA security checking is enabled, OTMA uses the RACF user ID for authorizing commands, transactions, and RESUME TPIPE calls with RACF.

When both a default RACF user ID is defined and the IRM_RACF_USERID field of an incoming message is not blank, IMS Connect uses the user ID defined in the IRM_RACF_USERID field.

Define the default RACF user ID in the RACFID parameter of the IMS Connect TCPIP configuration statement.

IMS Connect RACF user ID cache

IMS Connect can be configured to use a memory cache for RACF user IDs instead of issuing RACF requests for every transaction.

When IMS Connect is configured to use RACF security, the RACF user ID provided for each message must be validated before the message can be processed. This method of RACF security authentication can negatively affect the performance of IMS Connect. To improve performance without disabling RACF support, you can enable the IMS Connect RACF user ID cache. The cache stores previously verified RACF user IDs from all sessions. When possible, the cached information is passed to OTMA without the need for a new RACF verification request.

During system definition, the settings of the RACF user ID cache are configured with the TCPIP statement of the IMS Connect member of the PROCLIB data set (HWSCFGxx). Specifically, the cache settings are defined with the RACF, UIDCACHE, and UIDAGE parameters. The RACF parameter defines whether IMS Connect uses RACF authentication, the UIDCACHE parameter defines whether the RACF user ID cache is enabled, and the UIDAGE parameter specifies the default refresh interval for cached IDs.

You can also enable or disable the cache when IMS Connect is running with any of the following commands:

- The WTOR command **SETUIDC**
- The z/OS Modify command **UPDATE MEMBER**
- The type-2 command **UPDATE IMSCON TYPE(CONFIG) SET(UIDCACHE(ON | OFF))**

When IMS Connect is running, it automatically monitors the RACF Event Notification Facility (ENF) events associated with the cached user IDs. If the RACF ENF issues a type 71 event for the RACF CONNECT or REMOVE commands, or for an ALTUSER REVOKE command, IMS Connect automatically refreshes the user ID. IMS Connect issues event number 258 after automatically refreshing the specified ID in the cache.

You can also refresh specific user IDs manually with any of the following commands:

- The WTOR command **REFRESH RACFUID**
- The z/OS Modify command **UPDATE RACFUID**
- The type-2 command **UPDATE IMSCON TYPE(RACFUID) NAME(userid) OPTION(REFRESH)**

Related reference

IMS Connect UPDATE RACFUID command (Commands)
IMS Connect UPDATE MEMBER command (Commands)
REFRESH RACFUID command (Commands)
SETUIDC command (Commands)
UPDATE IMSCON TYPE(CONFIG) command (Commands)
UPDATE IMSCON TYPE(RACFUID) command (Commands)
Event types (Exit Routines)

IMS Connect security for clients of IMS DB

For clients that connect to IMS DB through ODBM, such as the IMS Universal drivers and clients using the Distributed Relational Database Architecture (DRDA), IMS Connect authenticates the user, but does not check the authority of the user to perform any actions.

To authenticate a user ID for an IMS DB client, IMS Connect can use the IMS Connect DB Security user exit routine (HWSAUTH0), a security product such as RACF, or both. For IMS DB clients, IMS Connect also provides support for RACF PassTickets.

IMS Connect always calls the HWSAUTH0 user exit, regardless of whether RACF or another security product is enabled. If RACF support is included in your IMS Connect configuration, IMS Connect calls the HWSAUTH0 user exit before invoking RACF.

If IMS Connect is configured to call RACF, you can enable RACF security statistics to be recorded when IMS Connect issues the RACF call **RACF RACROUTE REQUEST=VERIFY** to authenticate ODBM client connections to IMS DB. You can enable RACF statistics either by specifying ODRACFST=Y in the ODACCESS statement or by issuing the online IMS Connect command **UPDATE IMSCON TYPE(CONFIG)**. After you enable RACF statistics, the statistics are recorded by RACF no more than once per day to a system management facility (SMF) data set or log stream. The SMF data set or log stream that is used to record the RACF statistics is specified in the RACF configuration.

The HWSAUTH0 user exit routine can override the input user ID with a different user ID and can provide a RACF group ID to be authenticated further by IMS Connect.

The HWSAUTH0 user exit routine is a BPE type-1 user exit routine and is refreshable.

IMS Connect does not support Secure Sockets Layer (SSL) directly for clients that connect to IMS DB. To secure connections to IMS DB with SSL, use IBM z/OS Communications Server Application Transparent Transport Layer Security feature (AT-TLS). The use of AT-TLS is transparent to IMS Connect.

Related concepts

IMS Connect definition and tailoring (System Definition)

Related tasks

“Enabling RACF security checking in IMS Connect” on page 176
You can enable RACF security checking in IMS Connect either by specifying RACF=Y in the HWS configuration statement or by issuing an online IMS Connect command.

“Enabling RACF security statistics for IMS Connect” on page 177
If IMS Connect is configured to call RACF, you can enable RACF security statistics to be recorded and updated when IMS Connect issues the RACF call **RACROUTE REQUEST=VERIFY**. You can enable RACF statistics to be recorded and updated for ODBM client connections to IMS DB and for OTMA client connections to IMS TM. After you enable RACF statistics, the statistics are updated no more than once per day.

“RACF PassTicket for IMS Connect client connections to IMS DB” on page 194
You can use RACF PassTickets to authenticate IMS Connect client connections to IMS DB. PassTickets are an alternative to RACF passwords and password phrases and provide better security because PassTickets remove the need to send passwords and password phrases across the network in clear text.

Related reference

UPDATE IMSCON TYPE(CONFIG) command (Commands)

ODACCESS statement (System Definition)

Passing network security credentials through IMS Connect

If security credentials are entered from an application in a distributed network environment and the application uses the HWSSMPL0, HWSSMPL1, or HWSJAVA0 user message exit routine, you can enable the credentials to be passed through IMS Connect to IMS. You can also enable the distributed network security credentials to be passed from IMS through IMS Connect in IMS callout requests.

The network security credentials are sent from IMS Connect to IMS in the security-data section of the OTMA message prefix. The network security credentials, including the network user ID and the network session ID, can then be included in the IMS log records, such as X'01' and X'03', that contain information about the OTMA message prefix. If you enable IMS Connect to pass distributed network security credentials in synchronous callout messages initiated by the ICAL call of the IMS DL/I interface to applications that issue a RESUME TPIPE call, the security credentials are also passed in the security-data section of the OTMA prefix.

Restriction: Distributed network security credentials from DataPower, IMS Connect API, and SOAP Gateway clients are not supported by IMS Connect.

Passing distributed network security credentials from user-written IMS Connect client applications that use the HWSSMPL0 or HWSSMPL1 user message exits

To pass distributed network security credentials from user-written IMS Connect client applications that use either the HWSSMPL0 or the HWSSMPL1 user message exits, use IRM extensions in the IMS request message (IRM) header. Specify an ID of *NETSID* for an IRM extension that contains the network session ID and an ID of *NETUID* for an IRM extension that contains the network user ID.

After the message that contains the *NETSID* or the *NETUID* extension, or both, is passed to the HWSSMPL0 or the HWSSMPL1 user message exit, the user message exit builds the OTMA message prefix to contain the network security credentials.

Recommendation: If network security credentials are included in IMS Connect client input messages, enable the BPE External Trace facility for the IMS Connect Recorder Trace facility. If network security credentials are passed to IMS Connect, the size of both input and output messages to and from IMS Connect might be larger than 670 bytes and the BPE External Trace facility would be required to capture the data of the entire message.

Passing distributed network security credentials to applications that issue a RESUME TPIPE call

To enable IMS Connect to pass distributed network security credentials in synchronous callout messages initiated by the ICAL call of the IMS DL/I interface to applications that issue a RESUME TPIPE call, define the RESUME TPIPE call with the following field specifications in the IRM prefix. If the following field specifications are not defined, IMS removes the distributed network security credentials from the security-data section of the OTMA message prefix in the callout request.

IRM_ARCH
X'05' (IRM_ARCH5)

IRM_F6
X'80' (IRM_F6_NWSE)

Passing distributed network security credentials from client applications of the IMS TM resource adapter

To enable IMS TM resource adapter to pass network security credentials from a Java EE application that uses the HWSJAVA0 user message exit routine to IMS, you must configure and link to your application the Java Authentication and Authorization Service (JAAS) login module that is provided with IMS TM resource adapter. After you link your application to the JAAS login module, users must enter their security credentials when they invoke an IMS transaction for authentication by an external user account registry. The external user account registry can be any user account registry that is supported by WebSphere Application Server or WebSphere Liberty such as an LDAP server. After the credentials are successfully authenticated, IMS TM resource adapter sends the distributed credentials to IMS Connect by using the security-data section of the OTMA message prefix.

You can also enable IMS TM resource adapter to support network security credentials when IMS applications that run in IMS dependent regions make synchronous or asynchronous callout requests to external Java EE applications.

Network security credentials in synchronous callout messages

To enable IMS TM resource adapter to support network security credentials in synchronous callout messages, set the resumeTpipeNsc property of the `IMSActivationSpec` object to true.

Network security credentials in asynchronous callout messages

To enable IMS TM resource adapter to support network security credentials in asynchronous callout messages, you must call the `setResumeTpipeNSC(int resumeTpipeNSC)` method for the `IMSInteractionSpec` object and set the value of the `setResumeTpipeNSC` property to 1. If 1 is set for the `setResumeTpipeNSC` property, IMS TM resource adapter sets a flag byte in the OTMA message prefix that is sent to IMS to indicate that network security credentials should be included in the callout message.

Related concepts

"RESUME TPIPE/receive protocol" on page 323
IMS Connect clients use the RESUME TPIPE protocol to retrieve commit-then-send (CM0) output or synchronous callout requests from a tpipe hold queue in IMS.

Related tasks

"Retrieving synchronous callout requests with RESUME TPIPE" on page 204
When issuing a RESUME TPIPE call to retrieve synchronous callout requests, you can code the RESUME TPIPE call to retrieve only synchronous callout messages or both synchronous callout messages and asynchronous output.

Related reference

"Format of the returned network security segments" on page 247
The Network Session ID (NETSID) and Network User ID (NETUID) segments contain network security information and are returned to the clients when the clients issue RESUME TPIPE calls.

"Format of IRM extensions" on page 227
You can use IMS request message (IRM) extensions to send information from IMS Connect client applications without expanding the DSECT that maps the IRM.

"Explanation of OTMA security data fields" on page 884
The following information provides additional detail on the content of the security-data section of the message prefix.

Related information

Network security credentials and IMS TM resource adapter

Configuring Websphere Application Server for distributed network security credentials

Configuring Websphere Liberty for distributed network security credentials

Enabling IMS TM resource adapter to support network security credentials in callout messages

Securing IMS-to-IMS TCP/IP connections

To secure IMS-to-IMS TCP/IP connections, IMS Connect uses RACF PassTickets to establish one instance of IMS Connect as a trusted user of another instance of IMS Connect.

When a connection is first established, the instance of IMS Connect that sends messages generates a RACF PassTicket and passes it to the instance of IMS Connect that receives the messages. After the receiving IMS Connect instance successfully verifies the PassTicket with RACF, any messages received on the connection are considered to be from a trusted user and are not subject to additional security checking.

The sending IMS Connect instance generates the RACF PassTicket from values provided on the APPL and USERID parameters of the RMTIMSCON statement.

The receiving IMS Connect instance calls RACF to authenticate the user ID and confirm authority to access the application by using the PassTicket, application name, and user ID sent by the sending IMS Connect instance.

Recommendations:

- If RACF is not enabled in the receiving IMS Connect instance, do not configure the sending IMS Connect instance to generate PassTickets. The receiving IMS Connect instance does not perform security checking and ignores any PassTicket data that is sent when RACF=N. Creating a PassTicket on the sending side wastes processing resources.
- Do not use RACF PassTickets with non-persistent connections, because doing so incurs significant processing overhead. A new PassTicket is generated and sent each time a new connection is established.

IMS Connect supports RACF PassTicket security for both MSC and OTMA communications.

For MSC communications, each instance of IMS Connect can send and receive transaction messages and responses. To secure IMS-to-IMS TCP/IP connections for MSC, you must enable RACF support and define application names and user IDs in both IMS Connect instances. The application names and user IDs defined in one instance can be different from those defined in the other instance. PassTicket classes, application names, and user IDs must also be created in RACF at both z/OS installations.

To secure an IMS-to-IMS TCP/IP connection between two instances of IMS Connect:

1. For the sending IMS Connect instance, specify the security settings in the IMS Connect configuration member in the IMS.PROCLIB data set:
   - On the RMTIMSCON statement, specify an application name and a user ID on the following parameters:
     - APPL
     - USERID
   - If the connection is used for MSC, specify RACF=Y on the HWS statement to enable security checking for the required return connection.
2. The security administrator at the sending IMS installation must create a RACF PassTicket class entry to support the generation of the PassTicket by IMS Connect.

   The PassTicket class defined at the sending installation must have the same key as the PassTicket class entry defined at the remote installation.

   The following example defines a PassTicket class to RACF:

   ```
   SETROPTS CLASSACT(PTKTDATA)
   SETROPTS RACLIST(PTKTDATA)
   RDEF PTKTDATA APPLI2I SSIGNON(KEYMASKED(E001193519561977)) UACC(N)
   SETROPTS REFRESH RACLIST(PTKTDATA)
   ```
3. For the receiving IMS Connect instance, specify the security settings in the configuration member in the IMS.PROCLIB data set:
   - On the HWS statement, specify RACF=Y.
   - If the connection is used for MSC, specify an application name and a user ID on the following parameters on the RMTIMSCON to provide the necessary security data for the return connection:
     - APPL
     - USERID
4. The security administrator at the receiving installation must define the user ID, application name, and the PassTicket class to RACF.

   The key of the PassTicket must match the key used by the PassTicket class defined at the sending installation.

   The following example defines two user IDs (USER0001 and USER0002), a PassTicket class (APPLI2I), and application name (APPLI2I) to RACF. Because only USER0002 is given access in RACF to APPLI2I, only USER0002 can establish a trusted user connection.

   ```
   DELUSER USER001
   ADDUSER USER001 PASSWORD(USER0001) TSO(ACCTNUM(D1001) PROC(TPROC02))
   DELUSER USER002
   ADDUSER USER002 PASSWORD(USER0002) TSO(ACCTNUM(D1001) PROC(TPROC02))
   ```

```
SETROPTS CLASSACT(PTKTDATA)
SETROPTS RACLIST(PTKTDATA)
RDEF PTKTDATA APPLI2I SSIGNON(KEYMASKED(...)) UACC(N)
RDEF PTKTDATA APPLI2I.USER002 SSIGNON(KEYMASKED(...)) +
     UACC(N) APPLDATA('NO REPLAY PROTECTION')
SETROPTS REFRESH RACLIST(PTKTDATA)

SETROPTS CLASSACT(APPL)
SETROPTS RACLIST(APPL)
RDEFINE APPL APPLI2I UACC(N)
PE APPLI2I ACCESS(READ) CLASS(APPL) ID(USER0002)
SETROPTS RACLIST(APPL) REFRESH
RLIST APPL APPLI2I AU
```

IMS Connect security exit routine

If any IMS Connect user message exit routine performs security checking, you must provide a security exit routine or use the z/OS TCP/IP IMS Listener security exit routine (IMSLSECX).

IMS does not provide a sample security exit routine due to the many options available for security and the fact that most installations have their own specific security method. The call to RACF is performed by IMS Connect if RACF parameters are provided in the OTMA header when the message exit routine returns the message.

The name of the security exit routine called by HWSSMPL0, HWSSMPL1, HWSSOAP1, or HWSCSLO0 is IMSLSECX.

If you use HWSSMPL0 or HWSSMPL1, you can change the name of the security exit routine that they call by changing EXTRN IMSLSECX to a name of your choice. If you change the name of the security exit routine, you must define the security exit routine in the HWSSMPL0 or HWSSMPL1 user message exit.

You can also provide the name of the security exit routine called by HWSJAVA0 and define it in the HWSJAVA0 message exit routine.

IMS Connect security and the OTMARTUX user exit

The OTMA Resume TPIPE Security user exit (OTMARTUX) is not an IMS Connect exit routine, but it is one of two possible methods that you can use to secure messages queued on the OTMA asynchronous hold queue. The other method is to use an external security product, such as RACF. You can use the OTMARTUX user exit and an external security product each by itself or in combination.

The OTMARTUX user exit runs in the IMS control region and not in the IMS Connect address space.

Related tasks

“Securing messages on the asynchronous hold queue” on page 813
You can protect messages on asynchronous hold queues from unauthorized use of the RESUME TPIPE call by using either RACF, the OTMA Resume TPIPE Security user exit (OTMARTUX), or both.

Related reference

OTMARTUX: OTMA Resume TPIPE Security user exit (DFSYRTUX and other OTMARTUX type exits) (Exit Routines)

HWSSMPL0 and HWSSMPL1 security actions

The sample user message exits HWSSMPL0 and HWSSMPL1 always perform certain security actions and perform other security actions only when the IMSLSECX security exit is or is not called.

Security actions when HWSSMPL0 and HWSSMPL1 do not call IMSLSECX

The following tables define the action that HWSSMPL0 and HWSSMPL1 take when they do not call the security exit (IMSLSECX).

Table 6. USERID results if security exit not called

	USERID field present in IRM	IRM USERID field blank/ null	RACF parms results passed in OTMA security header
USERID	Yes	Yes	Default RACFID
USERID	Yes	No	IRM USERID
USERID	No	N/A	Default RACFID

Table 7. GROUPID results if security exit not called

	GROUPID field present in IRM	IRM USERID field blank/ null	RACF parms results passed in OTMA security header
GROUPID	Yes	Yes	Blanks/nulls
GROUPID	Yes	No	IRM GROUPID
GROUPID	No	N/A	Blanks/nulls

Table 8. Password results if security exit not called

	Password field present in IRM	IRM PASSWORD field blank/null	RACF parms results passed in OTMA security header
PASSWORD	Yes	Yes	Blanks/nulls
PASSWORD	Yes	No	IRM PASSWORD
PASSWORD	No	N/A	Blanks/nulls

Security actions when HWSSMPL0 and HWSSMPL1 call IMSLSECX

The following tables define the action that HWSSMPL0 and HWSSMPL1 take when they call the security exit (IMSLSECX).

Table 9. USERID results if security exit called; returns blank or non-blank USERID

	USERID field present in IRM	IRM USERID field blank/null	Security exit return USERID	RACF parms results passed in OTMA security header
USERID	Yes	Yes	No	Default RACF USERID
USERID	Yes	Yes	Yes	Security exit returned USERID
USERID	Yes	No	No	USERID passed in IRM
USERID	Yes	No	Yes	Security exit returned USERID
USERID	No	N/A	No	Default RACF USERID
USERID	No	N/A	Yes	Security exit returned USERID

Table 10. GROUPID results if security exit called; returns non-blank USERID

	GROUPID field present in IRM	IRM GROUPID field blank/null	Security exit return GROUPID	RACF parms results passed in OTMA security header
GROUPID	Yes	Yes	No	Blank GROUPID
GROUPID	Yes	Yes	Yes	Security exit returned GROUPID

Table 10. GROUPID results if security exit called; returns non-blank USERID (continued)

	GROUPID field present in IRM	IRM GROUPID field blank/null	Security exit return GROUPID	RACF parms results passed in OTMA security header
GROUPID	Yes	No	No	Blank GROUPID
GROUPID	Yes	No	Yes	Security exit returned GROUPID
GROUPID	No	N/A	No	Blank GROUPID
GROUPID	No	N/A	Yes	Security exit returned GROUPID

Important: If the security exit returns a blank USERID, as shown in the following table, then the GROUPID that is returned by the exit is not used.

Table 11. GROUPID results if security exit called; returns blank USERID

	GROUPID field present in IRM	IRM GROUPID field blank/null	Security exit return GROUPID	RACF parms results passed in OTMA security header
GROUPID	Yes	Yes	No	Blank GROUPID
GROUPID	Yes	Yes	Yes	Blank GROUPID
GROUPID	Yes	No	No	IRM GROUPID
GROUPID	Yes	No	Yes	IRM GROUPID
GROUPID	No	N/A	No	Blanks
GROUPID	No	N/A	Yes	Blanks

Security actions that HWSSMPL0 and HWSSMPL1 always perform

If an IRM contains IRM extensions that have an ID of *NETUID* or *NETSID*, HWSSMPL0 and HWSSMPL1 build the security-data section of the OTMA message prefix to contain the network security credentials that are in the IRM extensions.

The following table defines the actions that HWSSMPL0 and HWSSMPL1 also take regardless of whether the security exit (IMSLSECX) is called. The password is based on the IRM, not on the security exit.

Table 12. Password results regardless of whether security exit called

	PASSWORD field present in IRM	PASSWORD field blank/null	Security exit return PASSWORD	RACF parms results passed in OTMA security header
PASSWORD	Yes	Yes	N/A	Blanks/nulls
PASSWORD	Yes	No	N/A	IRM PASSWORD
PASSWORD	No	N/A	N/A	Blanks/nulls

IMS Connect responses to errors on RACROUTE calls from the sample exits

If an error occurs on a RACROUTE call from one of the sample user message exits, HWSSMPL0 or HWSSMPL1, IMS Connect decides what error actions to take depending on the RACF parameter setting in the IMS Connect configuration file, as well as the specific circumstances that cause the error.

The different actions IMS Connect takes are described in the following series of tables.

IMS Connect response when RACF=Y and RACROUTE call parameters are in error

The following table describes the error actions that IMS Connect takes if RACF=Y, based on required RACROUTE call parameters.

Table 13. IMS Connect error actions taken based on RACROUTE call parameters (RACF=Y)

USERID	PASSWORD	GROUPID	Action taken
Non-blanks	Non-blanks	Non-blanks	RACROUTE call issued
Non-blanks	Blanks	Blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNOPW' • Password cleared in OTMA header • * Security failed, no password *
Non-blanks	Blanks	Non-blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNOPW' • Password cleared in OTMA header • * Security failed, no password *
Blanks	Non-blanks	Blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNUID' • Password cleared in OTMA header • * Security failed, no password *
Blanks	Non-blanks	Non-blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNUID' • Password cleared in OTMA header • * Security failed, no password *

Table 13. IMS Connect error actions taken based on RACROUTE call parameters (RACF=Y) (continued)

USERID	PASSWORD	GROUPID	Action taken
Blanks	Blanks	Non-blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNPUI' • Password cleared in OTMA header • * Security failed, no password *
Blanks	Blanks	Blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNPUI' • Password cleared in OTMA header • * Security failed, no password *
Non-blanks	Blanks	Non-blanks	• Error message HWSP1503 issued • Input rejected • RACROUTE call not issued • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFNOPW' • Password cleared in OTMA header • * Security failed, no password *

IMS Connect responses when RACF=Y and either RACROUTE call fails or OTMA header data is in error

The following table describes the error actions that IMS Connect takes if RACF=Y, either based on OTMA header data, or should the RACROUTE call fail.

Table 14. IMS Connect actions taken for RACROUTE call failure or OTMA header data error and RACF=Y

RACROUTE call failure or OTMA header data	Action taken
No security header	• Error message HWSP1503 issued • Input rejected • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='NOSECHDR' • Password cleared in OTMA header • No RACF call made

Table 14. IMS Connect actions taken for RACROUTE call failure or OTMA header data error and RACF=Y (continued)

RACROUTE call failure or OTMA header data	Action taken
Security header < X'6A'	• Error message HWSP1503 issued • Input rejected • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='INVSECHL' • Password cleared in OTMA header • No RACF call made
Conversation continued	• No error message issued • Input accepted • Password cleared in OTMA header • No RACF call made
Response message	• No error message issued • Input accepted • Password cleared in OTMA header • No RACF call made
UTOKEN present	• No error message issued • Input accepted • Password cleared in OTMA header • No RACF call made
RACROUTE call failed	• Error message HWSP1500 issued • Input rejected • Set OMUSR_RETCODE=X'04' • Set OMUSR_RESCODE='SECFAIL' • Password cleared in OTMA header • RACF return/reason codes in HWSP1500 message
All others	See Table 13 on page 186

IMS Connect response when RACF=N and RACROUTE call parameters are in error

The following table describes the error actions that IMS Connect takes if RACF=N, based on required RACROUTE call parameters.

Table 15. IMS Connect error actions taken based on RACROUTE call parameters (RACF=N)

USERID	PASSWORD	GROUPID	Action taken
Non-blanks	Non-blanks	Non-blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA

Table 15. IMS Connect error actions taken based on RACROUTE call parameters (RACF=N) (continued)

USERID	PASSWORD	GROUPID	Action taken
Non-blanks	Blanks	Blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA
Non-blanks	Blanks	Non-blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA
Blanks	Non-blanks	Blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA
Blanks	Non-blanks	Non-blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA
Blanks	Blanks	Non-blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA
Blanks	Blanks	Blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA
Non-blanks	Blanks	Non-blanks	• Password cleared • Bypass RACROUTE call • Pass these parameters to OTMA

IMS Connect response when RACF=N and RACROUTE call fails or OTMA header data is in error

The following table describes the error actions that IMS Connect takes if RACF=N, either based on OTMA header data, or should the RACROUTE call fail.

Table 16. IMS Connect error actions taken for RACROUTE call failure or OTMA header data (RACF=N)

RACROUTE call failure or OTMA header data	Action taken
No security header	• Password cleared • Bypass RACROUTE call • Pass OTMA headers and data to IMS OTMA
Security header < X'6A'	• Password cleared • Bypass RACROUTE call • Pass OTMA headers and data to IMS OTMA

Table 16. IMS Connect error actions taken for RACROUTE call failure or OTMA header data (RACF=N) (continued)

RACROUTE call failure or OTMA header data	Action taken
Conversation continued	• Password cleared • Bypass RACROUTE call • Pass OTMA headers and data to IMS OTMA
Response message	• No error message issued • Input accepted • Password cleared in OTMA header • No RACF call made
UTOKEN present	• Password cleared • Bypass RACROUTE call • Pass OTMA headers and data to IMS OTMA
RACROUTE call failed	• Password cleared • Bypass RACROUTE call • Pass OTMA headers and data to IMS OTMA
All others	• Password cleared • Bypass RACROUTE call • Pass OTMA headers and data to IMS OTMA

IMS Connect password management

IMS Connect provides several features to help you manage RACF passwords. Some of these features only apply when IMS Connect is configured to call RACF directly.

Changing RACF passwords by using client messages

When IMS Connect is configured to call RACF directly, users of the user message exit routines HWSSMPL0, HWSSMPL1, and HWSJAVA0 can change RACF passwords by submitting a client message that includes a password change request keyword.

To enable this feature, you must bind the HWSPWCH0 object code with the user message exit routine you are using. The HWSPWCH0 object code is stored in the IMS.ADFSLOAD member of the distribution library.

The password change request keyword must appear at the beginning of the application data section of the message and be followed by a blank, the old password, the new password, and the new password again.

For example, for the HWSSMPL0 and HWSSMPL1 user message exit routines, a password change request message has the following format:

```
llllIRMllzzHWSPWCH old-password/new_password/new_password|llzz
```

For the HWSJAVA0 user message exit routine, a password change request message has the following format:

```
llllIRMOTMALLzzHWSPWCH old_password/new_password/new_password|llzz
```

The password change request keyword that is defined in the sample user message exit routines is "HWSPWCH." You can change this keyword by modifying the user message exit routine. Any password change request keyword that you define must be followed by a blank as a delimiter.

IMS Connect returns a response message to the client application in one of the following formats:

- For the user message exit routine HWSSMPL0, the format is:

```
llzzmessage_textCSMllzz
```

- For the user message exit routine HWSSMPL1, the format is:

```
llllmessage_textCSMllzz
```

- For the user message exit routine HWSJAVA0, the format is:

```
llllOTMAheadermessage_textllzz
```

The communication sequence for a password change request is:

1. Connect.
2. Send the password change request.
3. Receive message HWSC00*xxy*, where *xx* is the final two digits of the message number and *y* is the message type identifier.

   If you are using HWSSMPL0 or HWSSMPL1, the user-written client application receives the message HWSC00*xxy* appended with *CSM0KY*.

   If you are using HWSJAVA0, the IMS TM Resource Adapter client application receives the HWSC00*xxy* message in the application data portion of the OTMA header and the appropriate return and reason codes in the user data portion of the OTMA header.

Related reference

IMS TM Resource Adapter user message exit routine (HWSJAVA0) (Exit Routines)

User message exit routines HWSSMPL0 and HWSSMPL1 (Exit Routines)

Changing RACF password phrases by using client messages

When IMS Connect is configured to call RACF directly, users of the user message exit routines HWSSMPL0, HWSSMPL1, and HWSJAVA0 can change RACF password phrases by submitting a client message that includes the password phrase change request keyword.

To enable this feature, you must bind the HWSPWCH0 object code with the user message exit routine you are using. The HWSPWCH0 object code is stored in the IMS.ADFSLOAD member of the distribution library.

The password phrase change request is similar to a password change request and it uses the same keyword. Therefore, the password change request keyword must appear at the beginning of the application data section of the message and be followed by a blank, the old password phrase, a blank, the new password phrase, a blank, and the new password phrase again. The password phrases must be enclosed in single quotation marks. If a single quotation mark is part of a password phrase, a second single quotation mark must follow it. There must be at least one blank between the password phrases.

For example, for the HWSSMPL0 and HWSSMPL1 user message exit routines, a password change request message has the following format:

```
llllIRMllzzHWSPWCH 'old_ phrase' 'new_phrase' 'new_phrase'|llzz
```

For the HWSJAVA0 user message exit routine, a password phrase change request message has the following format:

```
llllIRMOTMAllzzHWSPWCH 'old_ phrase' 'new_phrase' 'new_phrase'|llzz
```

The password change request keyword that is defined in the sample user message exit routines is "HWSPWCH." You can change this keyword by modifying the user message exit routine. Any password change request keyword that you define must be followed by a blank as a delimiter.

IMS Connect returns a response message to the client application in one of the following formats:

For the user message exit routine HWSSMPL0, the format is:

```
llzzmessage_textCSMllzz
```

For the user message exit routine HWSSMPL1, the format is:

```
llllmessage_textCSMllzz
```

For the user message exit routine HWSJAVA0, the format is:

```
llllOTMAheadermessage_textllzz
```

The communication sequence for a password change request is:

1. Connect.
2. Send the password change request.
3. Receive message HWSC00*xxy*, where *xx* is the final two digits of the message number and *y* is the message type identifier. If you are using HWSJAVA0, the IMS TM Resource Adapter client application receives the HWSC00*xxy* message in the application data portion of the OTMA header and the appropriate return and reason codes in the user data portion of the OTMA header.

Enabling mixed-case password support

IMS Connect supports mixed-case passwords. To use mixed-case passwords, RACF (or your similar security product) must also support mixed-case passwords.

By default, IMS Connect support for mixed-case passwords is determined by the configuration of the security product in use. For example, if RACF supports mixed-case passwords, IMS Connect automatically supports mixed-case passwords also.

If IMS Connect cannot determine if the security product supports mixed-case passwords, IMS Connect translates all passwords to uppercase, unless instructed to do otherwise during system definition or by an online command while IMS Connect is running.

During IMS Connect system definition, you can enable, disable, or accept the RACF specification for mixed-case password support by using the PSWDMC parameter in the HWS configuration statement.

In the online system, you can enable, disable, or accept the RACF specification for mixed-case password support by using any of the following commands:

- The IMS Connect WTOR command **SETPWMC ON | OFF | RCF**
- The IMS Connect type-2 command **UPDATE IMSCON TYPE(CONFIG) SET(PSWDMC(ON | OFF | RCF))**
- The IMS Connect z/OS command **UPDATE MEMBER TYPE(IMSCON) SET(PSWDMC(ON | OFF | RCF))**

You can view the current setting for mixed-case password support by issuing any of the following commands:

- The IMS Connect WTOR command **VIEWHWS**
- The type-2 command **QUERY IMSCON TYPE(CONFIG)**
- The IMS Connect z/OS command **QUERY MEMBER**

Related reference

IMS Connect UPDATE MEMBER command (Commands)

IMS Connect QUERY MEMBER command (Commands)

SETPWMC command (Commands)
VIEWHWS command (Commands)
UPDATE IMSCON TYPE(CONFIG) command (Commands)
QUERY IMSCON TYPE(CONFIG) command (Commands)
HWS statement (System Definition)

IMS Connect support for RACF PassTicket

An alternative to the RACF password is a PassTicket. PassTicket allows you to communicate with a host without using a RACF password. When IMS Connect is configured to call RACF directly, you can use PassTicket to authenticate user IDs and log on to computer systems that contain RACF.

RACF PassTicket for IMS Connect client connections to IMS TM

An alternative to the RACF password is a PassTicket, which you can use for IMS Connect client access to IMS TM. PassTicket allows you to communicate with a host without using a RACF password. When IMS Connect is configured to call RACF directly, you can use PassTicket to authenticate user IDs and log on to computer systems that contain RACF.

For IMS Connect clients that access IMS TM, you can select PassTicket support through the client and send a PassTicket in the IRM in place of a RACF password. IMS Connect issues a RACF call using PassTicket and blanks out the PassTicket field in the OTMA user data header before sending the message to IMS. Because PassTicket occupies the same field as the RACF password and PassTicket cannot be translated to uppercase, the RACF password is also not translated to uppercase. You can use a user message exit to provide uppercase translation.

The IMS Connect PassTicket support for IMS TM clients parallels IMS PassTicket support.

- You can use existing APPLname definitions for newly connecting IMS Connect clients.
- Each DATASTORE statement has a parameter APPL=*APPLname*, where:
  - Each APPL= can be a unique RACF APPLname for each data store.
  - Each APPL= can be the same name for each data store, as required for VGR support, or can be unique per data store.
- The default APPL=*APPLname* value is blank.
- The IMS Connect client can pass an APPLname in the IRM to the user message exit which sets the APPLname in the OTMA user data header or the user message exit can pass and set the appropriate APPLname in the OTMA user data header.

For PassTicket support, you are responsible for all definitions to RACF. You need to establish the RACF encoding and decoding routines and to supply the encoding routine to the distributed platform.

For IMS TM clients, The IMS TM Resource Adapter does not generate RACF PassTickets. PassTickets which have been generated can be passed through IMS TM Resource Adapter by setting the PassTicket value as the password.

This support might require changes to the customer-written user message exits and customer-written client application code. The following list describes options you can select for PassTicket support for IMS Connect clients that access IMS TM:

- **Support for passing an APPLname in the IRM to IMS Connect**

  This support has been added to the IRM definition. A new 8 byte field, IRM_APPL_NM, has been added to the end of the IRM structure. If you want to implement the PassTicket function for IMS Connect client access to IMS TM, then the client code must pass the APPLname to IMS Connect in this field.

  Note: This **will** change the length of the IRM by 8 bytes and the total length of the message by 8 bytes.

  The supplied user message exits (HWSSMPL1 and HWSSMPL0) have been modified so that a client can send an APPLname to IMS Connect in the IRM_APPL_NM field.

If you choose this option, you need only to pass the APPLname in the IRM. HWSIMSCB and IMS Connect have been modified to support this function.

- **No support for passing an APPLname in the IRM to IMS Connect**

  This support has been added to the IRM definition. A new 8 byte field IRM_APPL_NM has been added to the end of the IRM structure. If you do not want to implement the PassTicket function for IMS Connect client access to IMS TM, you have two options:

  – **Option 1: Blank APPLname**

    You can choose to pass a blank APPLname to IMS Connect in the IRM_APPL_NM field to IMS Connect.

    Note: This **will** change the length of the IRM by 8 bytes and the total length of the message by 8 bytes.

    The supplied user message exits (HWSSMPL1 and HWSSMPL0) have been modified so that a client can send a blank APPLname in the IRM_APPL_NM field to IMS Connect.

    If you choose this option, you need only to pass a blank APPLname in the IRM. HWSIMSCB and IMS Connect have been modified to support this blank APPLname function.

  – **Option 2: No APPLname**

    The customer can choose to pass no APPLname to IMS Connect in the IRM_APPL_NM field to IMS Connect.

    Note: This **will not** change the length of the IRM or the total length of the message.

    The supplied user message exits (HWSSMPL1 and HWSSMPL0) have been modified so that a client does not have to send an APPLname in the IRM_APPL_NM field to IMS Connect.

    If you choose this option, you do not need to perform any action. HWSIMSCB and IMS Connect have been modified to support this function of not passing an APPLname.

The APPLname is always passed to RACF. This is true even if PassTickets are not used. As a result, the APPL= keyword on the DATASTORE statement can be used to verify a user's authority to access an IMS Connect data store, even if PassTickets are not used.

RACF PassTicket for IMS Connect client connections to IMS DB

You can use RACF PassTickets to authenticate IMS Connect client connections to IMS DB. PassTickets are an alternative to RACF passwords and password phrases and provide better security because PassTickets remove the need to send passwords and password phrases across the network in clear text.

When RACF PassTickets are used to authenticate user access from a DRDA client to IMS DB, the PassTickets can be generated by the SQL Batch utility. If you use another DRDA client instead of the SQL Batch utility to access IMS DB, you can use another method that uses the RACF PassTicket generator algorithm to generate and evaluate PassTickets for your DRDA client.

The following high-level process describes how an IMS Connect client connection to IMS DB is authenticated with a RACF PassTicket if the client uses DRDA:

1. When the client connection is first established, the RACF PassTicket that is used to authenticate the connection to IMS DB is generated either by the SQL Batch utility or, for other DRDA clients, by a service that uses the RACF PassTicket generator algorithm.
2. The client application sends to IMS Connect the generated PassTicket and the ID of the user requiring access in the **SECCHK** command (X'106E'). The PassTicket is specified in the code point, X'11A1', for the **PASSWORD** parameter of the **SECCHK** command. The user ID is specified in the code point, X'11A0', for the **USRID** parameter of the **SECCHK** command.
3. IMS Connect issues the RACROUTE REQUEST=VERIFY call to RACF to authenticate the client connection. On the RACF RACROUTE REQUEST=VERIFY call, IMS Connect includes the following information:

- The RACF PassTicket and the user ID sent from the client application in the **SECCHK** command (X'106E').
- The application name as specified on the **APPL=** parameter of the ODACCESS statement, which is in the HWSCFG*xx* member of the IMS PROCLIB data set. If an application name is not specified on the **APPL=** parameter of the ODACCESS statement, IMS Connect uses instead the value that is specified on the **ID=** parameter of the HWS statement, which is also in the HWSCFG*xx* member.

Tips:

- If RACF is not enabled in the IMS Connect instance, do not configure the DRDA client to generate PassTickets. The IMS Connect instance does not perform security checking and ignores any PassTicket data that is sent when RACF=N. Creating a PassTicket on the DRDA client side wastes processing resources.
- Do not use RACF PassTickets with non-persistent connections because doing so incurs significant processing overhead. A new PassTicket is generated and sent each time a new connection is established.

To secure connections from a DRDA client to IMS DB by using a RACF PassTicket, perform the following steps:

1. Define to RACF the PassTicket class, application profile, application name, and user ID:
   a) Activate the PTKTDATA class. The PTKTDATA class is the class to which all profiles that contain PassTicket information are defined. To activate the class and the function, enter the following command:

      ```
      SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
      ```

   b) Enter the following commands to define the name of the application that users require access to by using the PassTicket:

      ```
      RDEFINE APPL <applname> UACC(NONE)
      SETROPTS CLASSACT(APPL)
      SETROPTS GENERIC(PTKTDATA)
      ```

      Where:

      applname
      Is a 1- to 8-character name for the application.

   c) Enter the **RDEFINE** command to define a profile for the application that users can gain access to with the PassTicket. The profile associates a secret secured sign-on application key with an application.

      ```
      RDEFINE PTKTDATA <applname> SSIGNON(<key_description>(<key>))
      ```

      Where:

      applname
      Is the 1- to 8-character application name that you defined in step “1.b” on page 195.

      key_description
      Specifies the method RACF is to use to protect the secured signon application key in the RACF database on the host. You can specify one of the following values:

      KEYMASKED
      Masks the secured signon application key.

      KEYENCRYPTED
      Encrypts the secured signon application key.

      key
      The secured signon application key, which is a user-supplied, 16-character hexadecimal value (0 – 9 and A – F).

   d) Enter the **PERMIT** command to permit a user ID to the application:

```
PERMIT APPLNAME CLASS(APPL) ID(<userid>) ACCESS(UPDATE)
```

Where:

userid
The user ID that is permitted to access the application. If you are using the SQL Batch utility to generate the PassTicket, the user ID must be the z/OS user ID that is associated with the batch job.

e) If you are using the SQL Batch utility to generate the PassTicket, enter the following command to permit the application to use the RACF PassTicket Generator service:

```
RDEFINE PTKTDATA IRRPTAUTH.<applname>.* UACC(NONE)
PERMIT IRRPTAUTH.<applname>.* CLASS(PTKTDATA) ID(<userid>) ACCESS(UPDATE)
```

Where:

applname
Is a 1- to 8-character name for the application defined in step "1.b" on page 195.

userid
The user ID that is permitted to access the application. If you are using the SQL Batch utility to generate the PassTicket, the user ID must be the z/OS user ID that is associated with the batch job.

f) Refresh the PTKTDATA class and the to activate the changes by entering the following commands:

```
SETROPTS RACLIST(APPL) REFRESH
SETROPTS RACLIST(PTKTDATA) REFRESH
```

2. Ensure that **RACF=Y** is specified in the HWS statement of the HWSCFG*xx* member.
3. Use one of the following methods to specify in the **APPL=** parameter of the ODACCESS statement the application name that is defined to RACF in the PTKTDATA class. That is, specify in the **APPL=** parameter the application name that you defined in step "1.b" on page 195. The application name that is specified on the **APPL=** parameter is used, in addition to the user ID and the RACF PassTicket, by IMS Connect on the RACF call RACROUTE REQUEST=VERIFY to authenticate DRDA client connections to IMS DB.

   - Directly add the **APPL=** parameter to the ODACCESS statement of the HWSCFG*xx* member.
   - Issue the following IMS type-2 command:

     ```
     UPDATE IMSCON TYPE(CONFIG) SET(ODBMAPPL(applname))
     ```

     Where *applname* is the application name that is defined to RACF in the PTKTDATA class.

   If the **APPL=** parameter is not specified, the value of the **ID=** parameter of the HWS statement, which is in the HWSCFG*xx* member, is used instead by IMS Connect in the RACF call RACROUTE REQUEST=VERIFY. Therefore, if the **APPL=** parameter is not specified, ensure that the application name that is defined in the PTKTDATA class is specified instead on the **ID=** parameter of the HWS statement.
4. Generate the PassTicket for the DRDA client.

   If you use the SQL Batch utility to generate the PassTicket, specify the **applname** URL property of the `DriverManager.getConnection` method.

   If you do not use the SQL Batch utility to generate the PassTicket, see Generating and evaluating a PassTicket for information on other methods to generate and evaluate a PassTicket by using the RACF PassTicket generator algorithm.
5. To use the generated PassTicket to authenticate the user of the IMS Connect client with RACF, use the **SECCHK** command (X'106E') to send the user ID and the PassTicket to IMS Connect. In the **SECCHK** command (X'106E'), include the user ID in the code point, X'11A0', for the **USRID** parameter of the command and include the PassTicket in the code point, X'11A1', for the **PASSWORD** parameter of the command.

Related concepts

Using RACF PassTickets (System Administration)

Generating and evaluating a PassTicket

Related tasks

Connecting to an IMS database by using the JDBC DriverManager interface (Application Programming)

"Enabling RACF security checking in IMS Connect" on page 176

You can enable RACF security checking in IMS Connect either by specifying RACF=Y in the HWS configuration statement or by issuing an online IMS Connect command.

Related reference

SECCHK command (X'106E') (Application Programming APIs)

QUERY IMSCON TYPE(CONFIG) command (Commands)

UPDATE IMSCON TYPE(CONFIG) command (Commands)

SQL Batch utility (Database Utilities)

ODACCESS statement (System Definition)

RACF PassTicket for IMS Connect Client connections to IMS OM

You can use RACF PassTickets as an alternative to passwords to authenticate user IDs for IMS Connect client connections to IMS Operations Manager (OM) for command requests.

PassTickets are an alternative to RACF passwords and password phrases and provide better security because PassTickets remove the need to send passwords and password phrases across the network in clear text.

For PassTicket support, you are responsible for all definitions to RACF. You need to establish the RACF PassTicket generation routine and to supply the routine to the distributed platform. To use RACF PassTickets, specify the PassTicket in the IRM in place of the password and also specify the application name.

For IMS Connect API for Java clients, the ICON API does not generate RACF PassTickets. Clients are responsible for generating the PassTickets and setting it in the command request message by issuing the `TmInteraction.setRacfPassword()` method. Clients must also specify the application name by issuing `TmInteraction.setRacfApplName()`.

The following high-level process describes how an IMS Connect client connection to IMS OM is authenticated with a RACF PassTicket:

1. The client generates the RACF PassTicket and specifies it in the command request in the `IRM_RACF_PW` field. It also specifies the application name in the `IRM_APPL_NM` field.
2. The client creates a connection to IMS Connect and sends the command request with the generated PassTicket, user ID and application name to IMS Connect.
3. IMS Connect issues the `RACROUTE REQUEST=VERIFY` call to RACF to authenticate the user ID of the client connection. On the RACF `RACROUTE REQUEST=VERIFY` call, IMS Connect includes the following information:
   - The RACF PassTicket and user ID sent from the client.
   - The application name sent from the client.

   Note: The application name should be defined to the RACF `PTKTDATA` class. The user ID should be permitted access to this application name.

To use a RACF PassTicket from an IMS Connect client to IMS OM, you need to define the PassTicket class, application profile, application name and user ID in RACF. Perform the following steps:

1. Add a user ID.

```
ADDUSER <authUserId> PASSWORD(<authUserIdPassword>)
SETROPTS RACLIST(APPL PTKTDATA) REFRESH
```

2. Activate the APPL and the PTKTDATA class.

```
SETROPTS CLASSACT(APPL PTKTDATA)
SETROPTS RACLIST(APPL PTKTDATA) REFRESH
```

THE PTKDATA class is where all profiles that use PassTickets are defined.

3. Define the name of the application in the APPL class that users require access to.

```
RDEFINE APPL <applName> UACC(NONE)
```

4. Define the application name profile in the PTKTDATA class with the specified secured signon key.

   The client uses the secured signon key to generate the PassTicket. The server uses the secured signon key to authenticate the PassTicket.

```
RDEFINE PTKTDATA <applName> SSIGNON(KEYMASKED(<signonKey>))
```

5. Grant the defined user read access to the application profile in the APPL class. The user ID can then be authenticated with a PassTicket.

```
PERMIT <applName> CLASS(APPL) ID(<authUserId>) ACCESS(READ)
```

6. Refresh the APPL class and the PTKTDATA class to activate the changes.

```
SETROPTS RACLIST(APPL PTKTDATA) REFRESH
RLIST APPL <applName> AU
```

PassTicket replay protection considerations

You can bypass PassTicket replay protection, which you might do if you have multiple end-users sharing the same user ID.

If you have multiple users with the same user IDs, it is possible for them to request access to an application during the same time interval. In this situation, the same PassTicket is generated for different users. As a result, if PassTicket replay protection is not bypassed, the users will be using the same PassTicket and be denied access to the application. Bypassing the PassTicket replay protection allows the same PassTicket to be used by multiple users.

Similarly, if you are stress testing your system where there is no think time driving requests to IMS Connect and have numerous requests to the same application occurring in the same time interval, you may want to consider bypassing PassTicket replay protection. This option allows the same PassTicket to be used within a ten minute period.

You can specify NO REPLAY PROTECTION in the APPLDATA field of the PTKTDATA profile for one or more of the selected applications to allow the same PassTicket to be generated within a ten minute period.

For additional information about no replay options, see "Protecting General Resources" in the *z/OS Security Server RACF Security Administrator's Guide*.

SSL and IMS Connect

To set up Secure Socket Layer (SSL) on TCP/IP connections to IMS Connect, use IBM z/OS Communications Server Application Transparent Transport Layer Security (AT-TLS).

When AT-TLS is used, it decrypts the incoming messages in z/OS before sending them to IMS Connect on a TCP/IP port and encrypts the outgoing messages before sending them to an IMS Connect client. IMS Connect is unaware of whether AT-TLS or SSL is used. For more information, see Setting up AT-TLS SSL for IMS Connect (System Definition).

Trusted-user support for IMS Connect messages

When IMS Connect is configured to call RACF directly, you can modify your user message exit to treat specific messages as *trusted users*. When a message is classified as a trusted user, IMS Connect does not

call RACF to check security for that message, but instead passes the specified user ID to OTMA without authentication.

After bypassing IMS Connect security, messages that are classified to IMS Connect as a trusted user are still subject to any security checking that might be performed by IMS OTMA. OTMA and IMS do not recognize the IMS Connect trusted user classification.

The following IMS Connect user message exits support the trusted user function:

- HWSSMPL0
- HWSSMPL1
- HWSJAVA0
- User-written user message exits

To enable trusted user support:

- Select one or more IRM fields or an HWSJAVA0 OTMA header fields to contain the flags that identify an input message as a trusted user.
- For IRM fields or customer-written prefix fields, define the bytes and byte settings that represent the trusted user flag so that the definitions are unique to your system. IMS Connect does not define which flag bytes to set or what settings to use.
- Code the user message exit to read the field for the flag and, when the trusted user flag is found, to set the X'80' bit for the OMUSR_TRSTUSR EQU flag in the OMUSR_FLAG2 field in the OTMA user data section of the message that the exit passes back to IMS Connect.

The IRM header fields you can use to identify trusted user messages can include one or more of the following fields: PORTID, CLIENTID, USERID, TRANSACTION CODE fields, and user data.

For example, you might decide to add three one-byte fields in the IRM and to set different values in each field. The client application sets the flag. When the message is passed to the user message exit, the exit interrogates the three fields. If the fields identify the message as a trusted user, the user message exit passes a request to IMS Connect in the OTMA header to bypass the call to RACF.

Sample logic, which is commented out, is provided in both HWSSMPL0 and HWSSMPL1 and can be found by looking for the following comment lines:

```
*************************************************
**************TRUSTED USER SUPPORT***************
*************************************************
```

If you are using the HWSJAVA0 user message exit, the exit identifies trusted user messages by the existing data in OTMA headers fields such as OMUSR_DESTID (DataStore), OMUSR_ORIGIN (ClientID), OMUSR_PORTID (PortId), OMUSR_PASSTICKET (Password), or other message values.

Code the HWSJAVA0 user message exit to set the OMUSR_FLAG2 flag to OMUSR_TRSTUSR. When the OMUSR_FLAG2 flag is set to OMUSR_TRSTUSR, IMS Connect bypasses the RACF call.

Specifying an OTMA ACEE aging value in the IMS Connect configuration member

You can configure IMS Connect to pass accessor environment element (ACEE) aging values to OTMA. ACEE aging values ensure that OTMA refreshes cached RACF ACEEs at appropriate frequencies for your installation.

The ACEE contains user IDs and other security information. OTMA caches the ACEE to improve the performance of the security checking that OTMA performs. The ACEEs are used to validate the authority of the user IDs included with the input messages passed by IMS Connect to the transactions and commands being requested.

To ensure that this security information is correct, OTMA automatically refreshes an ACEE with the security information that is stored in RACF when z/OS notifies IMS that the security information was modified. However, OTMA also refreshes ACEEs at defined intervals, regardless of whether z/OS notifies

IMS of changes. The interval between refreshes of the ACEE is determined by the ACEE aging value, which is set in OTMA.

To define an ACEE aging value for IMS Connect to pass to OTMA, use the OAAV= keyword in the DATASTORE configuration statement of the IMS Connect PROCLIB member, HWSCFGxx. You can specify a value from 300 to 999999 seconds. The default is 999999 seconds. OTMA requires a value of at least 300 to enable ACEE refreshes. The aging value that is specified on the OAAV= keyword is overridden in the following situations:

- If you issue the **/SECURE OTMA ACEEAGE** command without using the **TMEMBER** parameter to specify an OTMA client.
- If you issue the **/SECURE OTMA ACEEAGE** command and specify IMS Connect by using the **TMEMBER** parameter.

Related reference

/SECURE command (Commands)

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

Chapter 14. IMS Connect support for callout requests

IMS Connect is a required component when IMS application programs issue callout requests through OTMA to data or service providers that are external to the IMS installation. For both types of callout request, IMS Connect serves as the TCP/IP gateway between the IMS Connect client that use TCP/IP and the OTMA component of IMS.

IMS Connect supports callout requests without requiring any changes to the IMS Connect configuration statements. You do, however, have to configure the IMS Connect clients to support callout requests.

You can use the following types of IMS Connect clients to support synchronous and asynchronous callout requests:

- IMS TM Resource Adapter
- IMS Enterprise Suite SOAP Gateway
- User-written IMS Connect client

The following IMS Connect exit routines support callout requests:

- HWSJAVA0
- HWSSOAP1
- HWSSMPL0
- HWSSMPL1

IMS provides various sample application programs to test callout support. For more information, see Samples for the callout function (Installation).

For information about how to configure IMS TM Resource Adapter and SOAP Gateway, see:

- Callout programming models (TM Resource Adapter)
- Enabling an IMS application as a web service consumer

Configuring user-written IMS Connect clients for synchronous callout requests

To support synchronous callout requests, user-written IMS Connect clients must be configured to retrieve new callout requests from IMS, acknowledge the receipt of the callout request (ACK or NAK), and to return the synchronous callout responses to IMS through IMS Connect.

Synchronous callout request messages are handled by IMS Connect and OTMA in much the same way as asynchronous output. That is, synchronous callout request messages are retrieved by issuing a RESUME TPIPE call. Many of the same rules and guidelines for retrieving asynchronous output also apply to retrieving synchronous callout request messages.

IMS provides various sample application programs to test callout support. For more information, see Samples for the callout function (Installation).

The high-level steps for configuring a user-supplied IMS Connect client to support synchronous callout messages include coding the client to:

1. Retrieve callout requests by using the RESUME TPIPE call.

   When retrieving new synchronous callout requests, a user-written IMS Connect client usually issues the RESUME TPIPE call with a very long timeout value or no timeout value. In the IRM for the RESUME TPIPE request, the client can optionally indicate that the client application supports control data from the ICAL callout message.

   If a synchronous callout request message is queued when the RESUME TPIPE call is received by IMS, OTMA sends it to the client through IMS Connect. If no callout message is queued, the IMS Connect

© Copyright IBM Corp. 1974, 2020

client waits in a receive state and OTMA sends the next callout request to the client immediately upon arrival at the output queue.

2. Acknowledge the successful (ACK) or unsuccessful (NAK) receipt of synchronous callout requests.

   To acknowledge the receipt or the rejection of a synchronous callout request, the user-written IMS Connect client must send an ACK or a NAK response to IMS to free the output queue for new callout messages. If an ACK or NAK is not received before the timeout value specified for either the synchronous callout request or the acknowledgment message itself, OTMA discards the callout request and frees the output queue.

3. Return a response message to IMS Connect by using the synchronous callout option of the send-only protocol. The response must include the correlator token that IMS Connect included in the original callout request.

   The response can be either the data requested by the IMS application or an error message.

In addition to these steps for configuring the IMS Connect client, the following additional steps are required to complete the configuration of your installation for synchronous callout support:

- Code an IMS application program to initiate a synchronous callout request by issuing the DL/I ICAL call. You can optionally add control data in the ICAL call to specify the URL for the port, user token, security information, or any other information.
- Code an OTMA destination descriptor to route the synchronous callout request to the IMS Connect client.

The following figure shows the flow of a synchronous callout request between IMS and a user-written IMS Connect client.

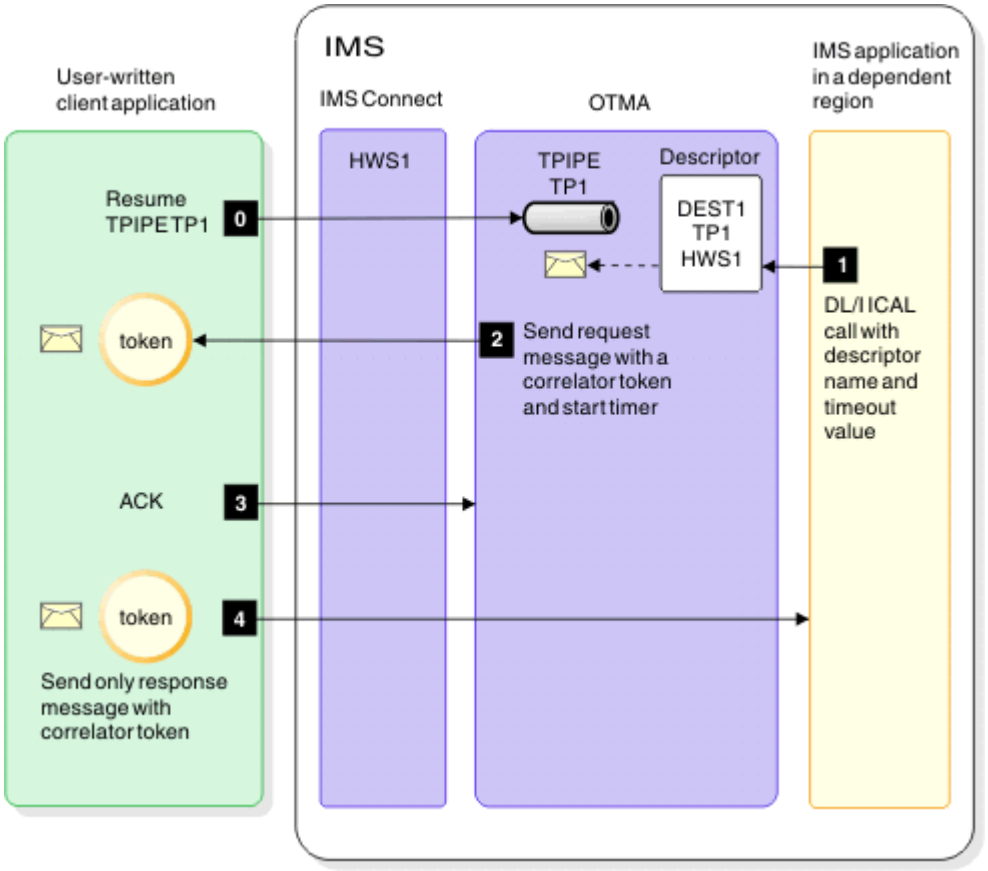


Figure 25. Flow of a synchronous callout request between an IMS application and a user-written IMS Connect client

Related tasks

"Retrieving synchronous callout requests with RESUME TPIPE" on page 204

When issuing a RESUME TPIPE call to retrieve synchronous callout requests, you can code the RESUME TPIPE call to retrieve only synchronous callout messages or both synchronous callout messages and asynchronous output.

Format of synchronous callout messages

Synchronous callout messages contain a segment for the correlator token. The correlator token is used to correlate responses with the IMS application programs that issue the callout requests. User-supplied client application programs must account for this token when reading callout request messages and return it with callout response messages.

The presence of the correlation token identifies the message to the IMS Connect client as a synchronous callout message. IMS generates the correlation token and includes it in the outgoing callout request automatically. The IMS Connect client must return the correlation token in the response that it sends back to IMS.

The application data segment in synchronous callout messages can be larger than 32 K and potentially many megabytes in length. IMS Connect includes the length of the application data segment at the beginning of the segment in a four-byte length field, LLLL. The length specified in the LLLL field includes the length of the LLLL field itself. The IMS Connect client must be configured to read all four bytes of the length field.

When the IMS Connect client returns the response to the callout request, the client must include the length of the application data segment in the four-byte length field at the beginning of the segment by using the same LLLL format.

Attention: Be aware that if multiple ICAL calls are issued concurrently, larger message sizes can consume larger amounts of extended private storage.

The following examples illustrate the format of synchronous callout requests and synchronous callout responses. In the examples, the segments enclosed in braces, { and }, are optional.

The structure of the correlation token itself cannot be modified; however, its structure is defined in the HWSIMSCB and HWSOMPFX macros.

The structure of a synchronous callout request message sent by IMS Connect to a client application conforms to the following format:

```
LLLL | LLZZ COR | {LLZZ RMM} | {Control data} | LLLL data | LLZZ CSM
```

The structure of a synchronous callout response returned by the client application must conform to the following format:

```
LLLL | LLZZ IRM | LLLL data | 00040000
```

For comparison, a message that is not a synchronous callout request and that does not support the 4-byte format of length field the application data segment looks like this:

```
LLLL | {RMM} | LLZZ data | {LLZZ data} | CSM
```

Related reference

“Output message from message exit to client” on page 239
Depending on the type of IMS Connect client and the user message exit used to support the client, the format of the message structure differs.

“Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines” on page 218
Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

“OTMA user data fields used by IMS Connect” on page 274

The format of the user data fields in the OTMA header is defined by the HWSOMUSR DSECT in the HWSOMPFX macro and is common to all IMS Connect messages.

"Format of the callout control data in the message" on page 246
The format of the callout control data in the output message that is sent to the client application depends on the type of client application that issues the resume tpipe request.

Retrieving synchronous callout requests with RESUME TPIPE

When issuing a RESUME TPIPE call to retrieve synchronous callout requests, you can code the RESUME TPIPE call to retrieve only synchronous callout messages or both synchronous callout messages and asynchronous output.

The RESUME TPIPE call for synchronous callout messages requires IRM architecture 3 (IRM_ARCH3) and a value of X'80' (IRM_F0_SYNONLY) or X'40' (IRM_F0_SYNASYN) in the IRM_F0 field. If no option is specified in the IRM_F0 field, the RESUME TPIPE call retrieves only asynchronous output.

IRM_F0_SYNONLY specifies that the IMS Connect client can only retrieve synchronous callout requests. If any asynchronous output messages are present on the tpipe hold queue when a RESUME TPIPE call that specifies IRM_F0_SYNONLY is received, the asynchronous messages are not returned to the client and remain on the tpipe hold queue.

IRM_F0_SYNASYN specifies that the IMS Connect client can retrieve both synchronous callout requests and any other asynchronous output messages on the queue. When IRM_F0_SYNASYN is specified, when a new RESUME TPIPE call is received, OTMA sends all synchronous callout requests first before sending asynchronous output messages.

If the RESUME TPIPE call from a client is connected to a different IMS Shared Queue member from the one that initiated and processed the synchronous callout request, the client will not receive the message even when the super member function is activated. The message is kept on the OTMA hold queue until timeout occurs and is deleted afterward. Because synchronous callout requests are queued to the tpipe hold queue, they are known only by the IMS that owns the tpipe. Super member function is honored for synchronous callout requests only when multiple IMS Connect clients are connected to the same IMS Shared Queue member.

- Define the RESUME TPIPE call for synchronous callout messages with the following field specifications in the IRM prefix:

  IRM_ARCH
  X'03' (IRM_ARCH3).

  IRM_F0
  Specify either:

  – X'80' (IRM_F0_SYNONLY) to retrieve only synchronous callout messages.

  – X'40' (IRM_F0_SYNASYN) to retrieve both synchronous callout messages and asynchronous output messages.

  IRM_F4
  R character value (IRM_F4_RESUMET).

  IRM_F5
  Retrieval option for the RESUME TPIPE call, such as X'02' (IRM_F5_AUTO).

  When the client application supports control data, also turn on the bit X'20' (IRM_F5_CTLDATA).

  IRM_F6
  If you require the client to receive network security credentials, specify X'80' (IRM_F6_NWSE) to indicate that the client supports the *NETSID* and *NETUID* output message segments.

  IRM_TIMER
  The timeout value for the RESUME TPIPE call, such as IRM_TIME_FF, which specifies that the RESUME TPIPE call does not time out.

Related concepts

"Managing the retrieval of output messages" on page 330
When retrieving either asynchronous output messages or synchronous callout request messages with the RESUME TPIPE call, you have options regarding how messages are returned.

"Timeout specifications on input messages" on page 311
Each and every input message from the IMS Connect client can set a different timeout value in the IRM_TIMER field of the fixed portion of the IMS request message (IRM) header.

"Socket connections" on page 300
IMS Connect provides three kinds of client TCP/IP connection protocols, which are called *sockets*. The TCP/IP sockets define how IMS Connect manages client TCP/IP connections when IMS Connect sends a disconnect message.

Related reference

"IMS Connect message structures" on page 213
TCP/IP clients using the z/OS program call interface communicate with IMS Connect by using an IMS request message (IRM) header on each input message. The IRM header is used on input messages from IMS Connect client application programs to communicate protocol options to IMS Connect. The IRM header is mapped by the IMS Connect HWSIMSCB macro.

"Output message from message exit to client" on page 239
Depending on the type of IMS Connect client and the user message exit used to support the client, the format of the message structure differs.

RESUME TPIPE error scenarios

If the IMS Connect client issues a RESUME TPIPE call for synchronous callout messages with the IRM_F0 field set to either IRM_F0_SYNONLY or IRM_F0_SYNASYN, but IRM_ARCH is not greater than or equal to IRM_ARCH3, IMS connect returns the following return and reason codes to the IMS Connect client in the RSM structure:

- RSM_RETCOD = RSMRTC_EXIT X'04'
- RSM_RSNCOD = RSMRSN_INVBUF X'09'

If IMS Connect retrieves a message on a tpipe that is used for synchronous callout message, but cannot deliver the message to the IMS Connect client, IMS Connect replies to OTMA with a NAK message. How OTMA handles the message after receiving the NAK message depends on the status of the IMS Connect client and whether the undeliverable message is a synchronous callout message or an asynchronous output message.

For asynchronous output messages, OTMA returns the asynchronous output message to the tpipe hold queue.

For synchronous callout messages:

- If IMS Connect could not deliver the synchronous callout message because the client either timed out or otherwise became disconnected, OTMA keeps the synchronous callout message on the queue for retrieval by another RESUME TPIPE call or until the timeout interval specified on the ICAL call expires.
- If IMS Connect could not deliver the synchronous callout message for a reason other than timeout or disconnection, such as TCP/IP errors, OTMA discards the synchronous callout message.

Acknowledging receipt of synchronous callout messages

A user-written IMS Connect client must send a positive (ACK) or negative (NAK) acknowledgment message to IMS after OTMA sends a synchronous callout request message. Until the ACK or NAK message is received or the timeout interval expires for either the acknowledgment or the original request, the tpipe queue remains in a wait state (WAIT_S) and cannot send or receive any other messages.

An ACK message frees the tpipe queue to send and receive other callout messages and the IMS application program continues to wait in the dependent region until either a response to the synchronous callout message is received or the synchronous callout request times out.

A NAK message not only frees the tpipe queue, but also tells OTMA what to do with the synchronous callout request message and whether to maintain or end the current RESUME TPIPE call.

The IMS Connect client can specify one of the following options on a NAK message:

- Discard the rejected synchronous callout request message and terminate the RESUME TPIPE connection.
- Discard the rejected synchronous callout request message, but maintain the RESUME TPIPE connection to continue retrieving other synchronous callout request messages.
- Keep the rejected synchronous callout request message on the tpipe queue, but terminate the RESUME TPIPE connection.

Related concepts

“Timeout intervals for IMS Connect acknowledgments to OTMA” on page 319
You can specify a timeout interval that determines how long OTMA waits for an acknowledgment from IMS Connect. You can also specify a timeout tpipe queue to hold commit-then-send (CM0) output after the timeout interval has expired.

Coding a NAK message to discard the callout message and end RESUME TPIPE call

To code a NAK message that directs OTMA to discard the rejected synchronous callout message and end the RESUME TPIPE call:

1. Specify N in the IRM_F4 field
2. Optional: Include an extended error code by specifying the following:
   - IRM_F0 = X'10'
   - IRM_NAK_RSNCDE = A two-byte hexadecimal extended error code

   The extended error code is returned to the IMS application in the AIBERRXT field of the AIB of the ICAL call.

Upon receiving this NAK message, OTMA issues to the IMS application program a return code of X'100' and a reason code of X'108'.

If the NAK is for an asynchronous output message, the asynchronous output is returned to the tpipe hold queue.

Coding a NAK message to discard a message, but keep a connection

To code a NAK message to direct OTMA to discard the rejected synchronous callout message, but keep the RESUME TPIPE call to retrieve other synchronous callout request messages:

1. Specify the following field values in the IRM of the NAK message:
   - IRM_F4 = N
   - IRM_F3 = X'08' (IRM_F3_REROUT)
2. Optional: Include an extended error code by specifying the following:
   - IRM_F0 = X'10'
   - IRM_NAK_RSNCDE = A two-byte hexadecimal extended error code

   The extended error code is returned to the IMS application in the AIBERRXT field of the AIB of the ICAL call.

Upon receiving this NAK message, the IMS application program receives a return code of X'100' and a reason code of X'108'.

If the NAK is for an asynchronous output message, the asynchronous output is returned to the reroute tpipe hold queue.

Coding a NAK message to retain message, but end RESUME TPIPE call

To code a NAK message to direct OTMA to keep the rejected synchronous callout message on the tpipe queue, but end the RESUME TPIPE call:

- Specify the following field values in the IRM of the NAK message:
  - IRM_F4 = N
  - IRM_F0 = X'20' (IRM_F0_SYNCNAK)

Upon receiving the NAK message, the RESUME TPIPE call ends, but OTMA keeps the synchronous callout request message on the tpipe queue. The rejected synchronous callout request message can then be retrieved by another RESUME TPIPE call, if the call is received by OTMA before the callout request times out.

If the NAK is for an asynchronous output message, OTMA places the message back on the hold queue and terminates the RESUME TPIPE call.

Returning callout responses to IMS

User-written IMS Connect clients return the callout response messages to IMS by using the send-only protocol for synchronous callout responses.

When you use the send-only protocol for synchronous callout responses, you can configure IMS to acknowledge receipt of the callout response, or you can suppress the acknowledgement for true send-only behavior.

The send-only message contains no transaction code and can contain either the response data or error information for the IMS application program.

A response to a synchronous callout request must be returned to the same IMS system in which the original DL/I ICAL call was issued. However, the response does not need to be returned by the same IMS Connect client that processed the original callout request. The response message can also be returned to IMS through a different IMS Connect instance than the instance that handled the outbound callout request. The correlation token included in the response message ensures that the response is returned to the correct IMS application program that is waiting for it.

The response message returned by IMS Connect client can be up to the length specified by the MAXSIZE parameter in the HWS configuration statement or the IMS Connect default of 10·MB.

The IMS Connect client must indicate in the IRM_F4 field of the message prefix that the message being returned is a synchronous callout response.

If the client requires IMS to return an acknowledgement after IMS receives the callout response, the client must specify an L in the IRM_F4 field of the IRM message prefix and switch to a receive state after sending the response.

If the client does not require IMS to return an acknowledgement, the client must specify an M in the IRM_F4 field.

If the client returns a synchronous callout response to IMS Connect, but either the IRM_ARCH is less than IRM_ARCH3 or the correlation token is not included in the IRM_CORTKN field, IMS Connect rejects the response and the synchronous callout request message times out.

When IMS acknowledges a callout response, IMS Connect returns a CSM (ACK) if delivery was successful and an RSM (NAK) if it was not successful. When the acknowledgements are disabled, IMS Connect does not return an RSM message to the client.

To code the callout response:

- Specify the following field values in the IRM of the response message:
  - IRM_ARCH = X'03' (IRM_ARCH3)
  - IRM_F4 = L (IRM_F4_SYNRESPA) or M (IRM_F4_SYNRESP)

– IRM_CORTKN = The 40-byte correlation token (CORTKN) that was included with the original callout request

For coding error responses for synchronous callout, see “Returning an error response to IMS” on page 208.

Related concepts

“Send-only protocol for synchronous callout responses” on page 299
IMS Connect clients return responses to synchronous callout requests from IMS application programs by using the send-only protocol.

Related reference

“IMS Connect message structures” on page 213
TCP/IP clients using the z/OS program call interface communicate with IMS Connect by using an IMS request message (IRM) header on each input message. The IRM header is used on input messages from IMS Connect client application programs to communicate protocol options to IMS Connect. The IRM header is mapped by the IMS Connect HWSIMSCB macro.

Returning an error response to IMS

If an error occurs after the user-written IMS Connect client has already returned an ACK message to IMS Connect and either the IMS Connect client or the data or service provider cannot complete the callout request, the IMS Connect client can return an error response to IMS instead of the expected data.

To code an error response to a DL/I ICAL call:

Specify the following field values in the IRM prefix of the error response:

- IRM_ARCH = X'03' (IRM_ARCH3)
- IRM_F0 = X'10' (IRN_F0_NAKRSN), if a NAK reason code is sent with the error response
- IRM_F0 = X'20' (IRM_F0_SYNCNAK), if no NAK reason code is sent with the error response
- IRM_NAK_RSNCDE = 2-byte hexadecimal extended error code
- IRM_F4 = L (IRM_F4_SYNRESPA), if the client requires IMS to return an acknowledgment
- IRM_F4 = M (IRM_F4_SYNRESP), if the client does not require IMS to return an acknowledgment
- IRM_CORTKN = the 40-byte correlation token generated by the original DL/I ICAL call.

Related concepts

“Send-only protocol” on page 296
Client application programs use the send-only protocol to submit commit-then-send (CM0) input messages to IMS in rapid succession without requiring the client application to wait for a response. The send-only protocol is designed for fast, high volume input. Any error that occurs in IMS Connect or IMS is not returned to the client.

Related reference

“IMS Connect message structures” on page 213
TCP/IP clients using the z/OS program call interface communicate with IMS Connect by using an IMS request message (IRM) header on each input message. The IRM header is used on input messages from IMS Connect client application programs to communicate protocol options to IMS Connect. The IRM header is mapped by the IMS Connect HWSIMSCB macro.

Chapter 15. IMS Connect XML message conversion

For some IMS Connect client application programs, IMS Connect can convert XML messages into either COBOL or PL/I, so that you do not need to modify existing IMS application programs to process messages submitted to IMS in XML.

IMS Connect conversion support converts the XML user data to the language structures expected by the IMS application program, which can then process the message as a normal IMS transaction message.

The IMS Connect XML conversion support is provided by an XML adapter function and an HWSSOAP1 message exit with a COBOL XML converter.

The actual conversion of XML messages is performed by an XML converter, which is called by IMS Connect.

IMS Connect supports XML conversion into the following languages for the following clients:

COBOL

- IMS Enterprise Suite SOAP Gateway
- IMS Web 2.0 Solution for IBM Mashup Center

PL/I

- IMS Web 2.0 Solution for IBM Mashup Center

IMS Connect also supports the conversion of multi-segment messages between SOAP Gateway and IMS application programs that function as Web service providers and that use multiple language structures. Restrictions apply to the multi-segment message support. These restrictions are detailed in the SOAP Gateway and IBM Developer for System z® documentation.

The IMS Connect HWSSOAP1 exit routine also supports XML conversion and you can specify an XML adapter name and an XML converter name in OTMA destination descriptors.

In the IMS request message (IRM) header of the input message, the IMS Connect client requests XML conversion support by specifying an XML adapter name and an XML converter name. IMS Connect reads the IRM when it receives an input message and, if XML conversion is required, IMS Connect calls the XML adapter. The XML adapter then calls the XML converter to perform the actual conversion. After the message is converted from its XML format, IMS Connect then sends the resulting message to IMS.

When IMS Connect receives the response message from IMS, IMS Connect calls the XML adapter, which in turn calls the XML converter to convert the user data in the response message into XML. IMS Connect then sends the output message to the IMS Connect client.

Using control data in synchronous callout messages to override the XML converter name: In outbound synchronous callout requests that are sent via SOAP Gateway, you can override the XML converter name that is used by IMS Connect. You can do so by having the IMS application program specify a different converter name in the control data area of the DL/I ICAL call when the application program issues the callout request. When the converter name is specified in the control data area, the name must be specified in uppercase EBCDIC characters and enclosed in the <DFSCVTNR> and </DFSCNVTR> tags.

Related tasks

Configuring XML conversion support for IMS Connect clients (System Definition)

Related reference

ICAL call (Application Programming APIs)

© Copyright IBM Corp. 1974, 2020

IMS Connect XML converters

When you enable IMS Connect XML conversion support, you must create the XML converters that are used by IMS Connect to convert the data structures from XML to the programming language used by the IMS application program and vice versa.

The XML converters are application programs. You can use the separately licensed tool IBM Developer for System z to automatically generate the XML converters from the COBOL copybook or the PL/I source code.

Recommendation:

For COBOL application programs, the XML converter is based on the COBOL copybook of the IMS COBOL application program that processes the message. For PL/I application programs, the XML converter is based on the source of the PL/I application program. Each IMS application that processes messages converted from XML must have its own unique XML converter.

The XML converters run in an IBM Language Environment® for z/OS enclave in the IMS Connect region. The amount of space required depends on the size and number of XML converters required in your environment. The IMS Connect region size must be increased to accommodate this storage requirement.

You can specify the maximum number of XML converters that this instance of IMS Connect can load concurrently via the MAXCVRT configuration parameter specified in the configuration member HWSCFGxx of the IMS PROCLIB data set. When your applications exceed the maximum number of XML converters specified in MAXCVRT, IMS Connect unloads the Least Recently Used XML converters first.

XML converters are defined as BPE exit list members. After initially creating an XML converter, you can update and refresh it without restarting IMS Connect by using any of the following commands:

- The WTOR command **REFRESH CONVERTER**
- The z/OS Modify command **UPDATE CONVERTER**
- The type-2 command **UPDATE IMSCON TYPE(CONVERTER) NAME(converter_name) OPTION(REFRESH)**

Related tasks

Configuring XML conversion support for IMS Connect clients (System Definition)

Related reference

ADAPTER statement (System Definition)

Structure of the XML message

An XML schema defines the XML tags that correspond to the COBOL data structures used by the COBOL IMS application program. The XML schema used for XML conversion is based on the COBOL copybook of the COBOL IMS application program.

The XML schema is not required by IMS Connect, but the application programmer that develops the web service that generates the XML input messages needs it.

If you use IBM Developer for System z to automatically generate the XML converters from the COBOL copybook of the IMS application program, IBM Developer for System z also generates the XML schema for you.

The XML tags defined by the XML schema directly correspond to the fields of the COBOL data structure.

The following is an example of an input and output message data structure defined in the COBOL copybook of the sample phone book application program that is available at the IMS Enterprise Suite SOAP Gateway download site through www.ibm.com/software/data/ims/soap/.

```
01  INPUT-MSG.
    02  IN-LL          PICTURE S9(3) COMP.
    02  IN-ZZ          PICTURE S9(3) COMP.
    02  IN-TRCD        PICTURE X(10).
    02  IN-CMD         PICTURE X(8).
    02  IN-NAME1       PICTURE X(10).
```

```
            02  IN-NAME2     PICTURE X(10).
            02  IN-EXTN      PICTURE X(10).
            02  IN-ZIP       PICTURE X(7).

        01  OUTPUT-MSG.
            02  OUT-LL       PICTURE S9(3) COMP.
            02  OUT-ZZ       PICTURE S9(3) COMP.
            02  OUT-MSG      PICTURE X(40).
            02  OUT-CMD      PICTURE X(8).
            02  OUT-NAME1    PICTURE X(10).
            02  OUT-NAME2    PICTURE X(10).
            02  OUT-EXTN     PICTURE X(10).
            02  OUT-ZIP      PICTURE X(7).
            02  OUT-SEGNO    PICTURE X(4)
```

Each field in the copybook has an equivalent XML tag that represents the field in the XML message. XML tags are case-sensitive. The dash symbol '-' in field names in the copybook is represented as an underscore '_' in the corresponding XML tags. The SOAP Gateway client has to build the XML message using that XML schema.

For example, the `IN-TRCD` field from the above copybook is represented in XML by the opening and closing tags <in_trcd> and </in_trcd>. In the data structure input message, the value of `IN-TRCD` should be placed in the byte positions 5 to 14. In the XML input message, the same value of `IN-TRCD` should be placed between the tags <in_trcd> and </in_trcd>.

An input messages from an IMS Connect client that uses the above example COBOL copybook of the phone book application has the following XML tags for the equivalent COBOL data structure fields:

```
<INPUTMSG>
<in_ll> </in_ll>
<in_zz> </in_zz>
<in_trcd> </in_trcd>
<in_cmd> </in_cmd>
<in_name1> </in_name1>
<in_name2> </in_name2>
<in_extn> </in_extn>
<in_zip> </in_zip>
</INPUTMSG>
```

The above COBOL copybook of the phone book application has the following XML tags for the equivalent outbound COBOL data structure fields:

```
<cbl:OUTPUTMSG>
<out_ll> </out_ll>
<out_zz> </out_zz>
<out_msg> </out_msg>
<out_cmd> </out_cmd>
<out_name1> </out_name1>
<out_name2> </out_name2>
<out_extn> </out_extn>
<out_zip> </out_zip>
<out_segno> </out_segno>
</cbl:OUTPUTMSG>
```

The input message XML tags must be wrapped by opening and closing XML tags based on the `01 INPUT-MSG.` definition in the COBOL copybook. In the above example, the opening and closing tags are <INPUTMSG> and </INPUTMSG>.

The output message XML tags generated by the outbound converter are wrapped by opening and closing XML tags that correspond to the `01 OUTPUT-MSG.` definition. In the above example, the opening and closing tags are <cbl:OUTPUTMSG> and </cbl:OUTPUTMSG>.

Look at the XML schemas to determine what these tags should be for each XML converter.

For each XML message, not all tags have to be specified, just like not all fields are required in the data structure message. The required fields are determined by the COBOL application.

Message conversion example

The code in this example describes the conversion of an XML input message to COBOL and its COBOL response message to XML.

Below is an example of an XML message on input:

```
<INPUTMSG><in_ll>32</in_ll><in_zz>0</in_zz>
<in_trcd>IVTNO</in_trcd><in_cmd>DISPLAY</in_cmd>
<in_name1>LAST1</in_name1></INPUTMSG>
```

Below is an example of the same message after the XML data structure has been converted to the COBOL data structure required by the COBOL phone book application program:

```
    IVTNO     DISPLAY LAST1
```

Note: In the above example, IVTNO starts in the fifth byte. The first four bytes are used by IMS Connect, and there are five spaces following LAST1.

The first four bytes of the data structure message, known as LLZZ, will be filled by the XML adapter. The first two bytes, the LL part, will be filled with the length of the data structure message. The second two bytes, the ZZ part, will be filled with zeroes. The XML converter converts the XML message to the application-specific format by taking the value within each XML tag, and placing it in its corresponding field position. The converted message can then be processed by the COBOL application and it returns an output message in its specific data structure. The output message has to be converted to XML before it is returned back to the Client. Below is an example of an output message and the message after XML conversion.

Below is an example of the COBOL data structure of the reply message on output from the IMS data store:

```
    ENTRY WAS DISPLAYED                    DISPLAY LAST1     FIRST1    8-111-1111D01/R010001
```

In the above example, the text on the first line of the output message begins at the fifth byte, after the four byte LLZZ field.

Below is an example of the same message after the COBOL data structure has been converted to XML as required by the IMS Connect client:

```
<cbl>

<out_ll> 093</out_ll><out_zz> 000</out_zz>
<out_msg>ENTRY WAS DISPLAYED</out_msg><out_cmd>DISPLAY</out_cmd>
<out_name1>LAST1</out_name1><out_name2>FIRST1</out_name2>
<out_extn>8-111-1111</out_extn><out_zip>D01/R01</out_zip>
<out_segno>0001</out_segno></cbl>
```

In the example above, the values following the opening XML tags <out_ll> and <out_zz> tags are from the first four bytes of the output COBOL data structure message.

For outbound messages from the IMS application, the conversion is performed in reverse. Each field in the application data structure is wrapped in its corresponding XML tags. This conversion from an XML to a COBOL application data structure message format, and vice versa, is performed by the COBOL XML converter called by the XML adapter.

Chapter 16. IMS Connect message structures

TCP/IP clients using the z/OS program call interface communicate with IMS Connect by using an IMS request message (IRM) header on each input message. The IRM header is used on input messages from IMS Connect client application programs to communicate protocol options to IMS Connect. The IRM header is mapped by the IMS Connect HWSIMSCB macro.

IMS Connect communicates with OTMA through an z/OS cross-system coupling facility session using the OTMA message headers. One of the primary tasks of the client-specific IMS Connect user-message exit routines is to translate the IRM header into the OTMA header on input.

Clients that use TCP/IP socket calls as their communication vehicle can design a user message exit routine that runs with IMS Connect to convert messages between formats as follows:

- Convert the client message format to OTMA message format.
- Convert the IMS response, in OTMA message format, to client message format.
- Convert XML in the user data portion of messages to COBOL and back again.

These conversions enable the client to retrieve IMS data through a TCP/IP connection. IMS Connect automatically sends and receives messages when they are formatted correctly.

Depending on the type of IMS Connect client and the user message exit used to support the client, the format of the IRM message header differs.

The following user message exit routines require specific message formats:

- HWSSMPL0 and HWSSMPL1
- HWSJAVA0
- HWSCSLO0 and HWSCSLO1
- HWSSOAP1

Related concepts

“Overview of IMS Connect exit routines” on page 159
IMS provides a variety of exit routines to support IMS Connect.

Related reference

IMS Connect exit routines (Exit Routines)

IRM structures for IMS Connect client messages

IMS Connect expects all client messages that it receives to start with a four byte total length field, followed by an IMS request message (IRM) header, followed by the message data segments.

After the four-byte length field, all IRMs begin with a 28 byte fixed-format section that is common to all messages from all IMS Connect client applications. You cannot modify the format of the fixed section of the IRM.

User-written client applications can include a user-defined section in the IRM after the fixed-format section. The user-defined section of the IRM allows you to specify additional options. If you add or delete any fields in the user-defined section of the IRM, you must also adjust the user message exits that you use, or provide your own user message exits. The user-defined section of the IRM is only supported by the HWSSMPL0 and HWSSMPL1 user message exit routines.

After the user-defined section of the IRM, you can define IRM extensions. IRM extensions enable information to be sent from an IMS Connect client application to IMS without expanding the DSECT that maps the IRM.

If you are using either the HWSSOAP1 user message exit, which is not provided as source, or the HWSJAVA0 user message exit, you cannot change the user portion of the IRM.

© Copyright IBM Corp. 1974, 2020

Note: The user message exits HWSCSLO0 and HWSCSLO1 do not have any message definitions because HWSCSLO0 and HWSCSLO1 cannot be modified or replaced.

Format of fixed portion of IRM in messages sent to IMS Connect

The IMS request message (IRM) header contains a 28-byte fixed-format section that is common to all messages from all IMS Connect client applications that communicate with IMS TM.

The following table shows the fixed format preceding the input message sent to IMS Connect from clients. It includes the message field, the field length, and a brief explanation of the message.

The llll field tells IMS Connect how long the message is, and the IRM provides additional information, such as the specific user exit to which the data is to be passed.

Table 17. Fixed portion of IRM prefix

Field	Length	Offsets		Meaning
		Dec	Hex	
llll	4 bytes	0	0	Length of the total message read as a binary number. The total message length includes the length of the llll field (four bytes), the IRM (variable length, depending on your requirements), the length of the message, and the end of message indicator X'00040000' (four bytes). The minimum length for all messages is X'58'. For IMS TM Resource Adapter, the maximum length is X'00989680' (10,000,000 bytes). For user-written client applications, the maximum length is X'7FFFFFFF'.
The following fields are for the 28 (decimal) byte IRM prefix.				
IRM_LEN	2 bytes	4	4	Length of the IRM structure. The minimum size of the IRM for user written exits is X'24' or binary '00100100'. HWSSMPL0 has a minimum IRM length of X'50' or binary '01010000'.

Table 17. Fixed portion of IRM prefix (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_ARCH	1 byte	6	6	Specifies the architectural level of the IRM prefix in messages received by IMS Connect from the client. • X'00' Specifies IRM_ARCH0, the base architectural structure of the user portion of the IRM prefix. • X'01' Specifies IRM_ARCH1, the architectural structure of the user portion of the IRM prefix that includes space for: – The IRM_REROUT_NM field – The IRM_RT_ALTCID field • X'02' Specifies IRM_ARCH2, the architectural structure of the user portion of the IRM prefix that includes space for: – The IRM_REROUT_NM field – The IRM_RT_ALTCID field – The IRM_TAG_ADAPT field – The IRM_TAG_MAP field • X'03' Specifies IRM_ARCH3, the architectural structure of the user portion of the IRM prefix that includes space for all of the fields included in IRM_ARCH2 and the following additional fields: – The synchronous callout correlation token fields – The IRM_MODNAME field for MFS MOD names • X'04' Specifies IRM_ARCH4, the architectural structure of the user portion of the IRM prefix that includes space for all of the fields included in IRM_ARCH3 and the field IRM_SESTKN for session tokens for IMS-to-IMS connections. • X'05' Specifies IRM_ARCH5, the architectural structure of the user portion of the IRM prefix that includes space for all of the fields included in IRM_ARCH4 and the following additional fields: – The IRM_EXTN_OFF field – A 2-byte reserved field

Table 17. Fixed portion of IRM prefix (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_F0	1 byte	7	7	• X'80' Specifies IRM_F0_SYNONLY: This is a resume tpipe call that retrieves only synchronous callout messages from IMS application programs running in IMS dependent regions. • X'40' Specifies IRM_F0_SYNASYN: This is a resume tpipe call that retrieves both synchronous callout messages from IMS application programs and asynchronous messages. • X'20' Specifies IRM_F0_SYNCNAK: This is a NAK message from the IMS Connect client that directs OTMA to keep the message that triggered the NAK on the tpipe queue. • X'10' Specifies IRM_F0_NAKRSN: This is a NAK message that includes a reason code for the NAK response. • X'04' Specifies IRM_F0_EXTENS, which indicates that the message contains one or more IRM Extensions. • X'01' Specifies IRM_F0_XMLTD, which indicates a request from an IMS Enterprise Suite SOAP Gateway client to convert an XML tagged message that contains both a transaction code and data into the format expected by the IMS application program. • X'02' Specifies IRM_F0_XML_D, which indicates a request from an SOAP Gateway client to convert an XML tagged message that contains data only into the format expected by the IMS application program.
IRM_ID	8 bytes	8	8	Character string. Specifies the identifier of the user exit that is to be driven after the complete message has been received. Additionally, IMS Connect reads this field to determine whether the incoming message is in ASCII or EBCDIC. The IMS Connect-supplied user message exits reserve and use these IDs: • *HWSCSL*-- for HWSCSLO0 • *HWSJAV*-- for HWSJAVA0 • *HWSOA1*-- for HWSSOAP1 • *SAMPL1*-- for HWSSMPL1 • *SAMPLE*-- for HWSSMPL0
IRM_NAK_RSNCDE	2 bytes	16	10	Optional reason code for a NAK response.
IRM_RES1	2 bytes	18	12	Reserved for future use. Initialize to binary zeros.

Table 17. Fixed portion of IRM prefix (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_F5	1 byte	20	X'14'	Input message type. **X'80'** OTMA headers built by client. **X'40'** Translation done by client. **X'20'** The client application that issues resume tpipe supports control data in the ICAL callout message. **X'10'** Single message with wait option. Only one message returned following the RESUME TPIPE call. If no message is present, OTMA waits for a message to arrive and then sends that single message to IMS Connect. The timer set on the RESUME TPIPE call can expire before a message is returned to IMS Connect. If that occurs, IMS Connect issues a NAK response to the message when received. **X'04'** No auto flow of messages. All current messages are returned one at a time. Use the no auto flow option only if the client is a dedicated output client. This value is similar to Auto, except that the IRM_TIMER will cause the last receive to terminate. Set the IRM_TIMER to a small value. Each ACK sent by the client resets the IRM_TIMER value. The IRM_TIMER value set by the RESUME_TPIPE only applies to the first receive state. **X'02'** Auto flow of messages. All current messages are returned one at a time. Use the auto flow option only if the client is a dedicated output client. Set the IRM_TIMER to a large value. Each ACK sent by the client resets the IRM_TIMER value. The IRM_TIMER value set by the RESUME_TPIPE only applies to the first receive state. **X'01'** Single message. Only one message returned following the RESUME TPIPE call. If no message is present, OTMA does not wait for a message and the IMS Connect timer causes a timeout to occur based on the timeout value specified. **X'08'** IRM_F5_XID The message includes an X/Open identifier (XID). **X'00'** No option flow of messages (see meaning for X'04'). This is the default if no value is specified.
IRM_TIMER	1 byte	21	X'15'	Time delay that IMS Connect will wait for IMS to return data to IMS Connect which, in turn, will be sent to the client. The following functions support the IRM_TIMER settings: • TCP/IP SEND of a RESUME TPIPE call • TCP/IP SEND of an ACK or NAK message • TCP/IP SEND of data • PC SEND of a RESUME TPIPE call • PC SEND of an ACK or NAK message • PC SEND of data

Table 17. Fixed portion of IRM prefix (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_SOCT	1 byte	22	X'16'	Socket connection type. The client can set this value as follows: **X'00'** Transaction socket. The socket connection lasts across a single transaction. **X'10'** Persistent socket. The socket connection lasts across multiple transactions. **X'40'** Non-persistent socket. The socket connection lasts for a single exchange consisting of one input and one output. **Recommendation**: Do not use this socket type if you plan on implementing conversational transactions, because multiple connects and disconnects will occur.
IRM_ES	1 byte	23	X'17'	Unicode encoding schema. Initialize to binary zeros. **X'01'** UTF8 encoding schema. **X'02'** UCS2 encoding schema. **X'02'** UTF16 encoding schema.
IRM_CLIENTID	8 bytes	24	X'18'	A string of 1 to 8 uppercase alphanumeric (A through Z, 0 to 9) or special (@, #, $) characters, left justified, and padded with blanks. IRM_CLIENTID specifies the name of the client ID that is used by IMS Connect. If this string is not supplied by the client, then the user exit must generate it. The client ID is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_CLID.

Related concepts

"Timeout specifications on input messages" on page 311
Each and every input message from the IMS Connect client can set a different timeout value in the IRM_TIMER field of the fixed portion of the IMS request message (IRM) header.

Related reference

"Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines" on page 218
Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines

Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

Messages submitted by the IMS Connect client might or might not include application or IMS data after the IRM depending on the type of message. For example, the following message types, which are specified in the IRM_F4 field, do not contain a data element after the IRM:

- ACK (A)
- CANCEL TIMER (C)

- DEALLOCATE (D)
- NAK (N)
- RESUME TPIPE (R)

Message types that do contain application data after the IRM include the following types, which are also specified in the IRM_F4 field:

- SENDONLY (S)
- SENDONLYA (K)
- SNDONLYE (J)
- Send-receive (X'40')
- Synchronous callout response with ACK (L)
- Synchronous callout response (M)

The data that follows the user portion of the IRM must be in the format of LLZZDATA, where LL is the total length of message segment (including the LL field), ZZ is binary zeros, and DATA is the IMS transaction code followed by the transaction data.

Restriction: If you are using IMS Connect client applications such as IMS TM Resource Adapter, IMS Enterprise Suite SOAP Gateway, or IBM Management Console for IMS and Db2 for z/OS, you cannot include a user-defined portion in the IRM.

The following table shows the format of the user portion of the IRM used by HWSSMPL0, HWSSMPL1, and other user-written user message exits.

This topic contains Product-sensitive Programming Interface information.

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits

Field	Length	Offsets		Meaning
		Dec	Hex	
llll	4 bytes	0	X'00'	Length of entire message including the IRM, application data, end of message indicator, and this length field
Fixed-format IRM	28 bytes	4	X'04'	The common, fixed portion of the IRM prefix.
IRM_MAP	0 bytes	32	X'20'	Position of the XID map when it is present.

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_F1	1 byte	32	X'20'	The purpose of this field differs depending on its value. **X'80' - IRM_F1_MFSREQ** The user requests that IMS return an MFS MOD name. **X'40' - IRM_F1_CIDREQ** The user requests that IMS Connect returns the client_id. **X'20' - IRM_F1_UC** This is a Unicode message. **X'10' - IRM_F1_UCTC** This is a Unicode transaction code. **X'04' - IRM_F1_SOARSP** No message text is returned with ACK for send-only with ACK requests **X'02' - IRM_F1_NOWAIT** This CM0 send-and-receive message uses the NOWAIT option for the expected ACK or NAK response. **X'01' - IRM_F1_TRNEXP** The user requests that IMS Connect set the expiration time for the input transaction. **X'00'** The user requests that no MFS MOD name to be returned. If this value is not supplied by the client, the user exit must use a default value. The MFS MOD name flag is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_UFLAG1. The Unicode encoding schema (UTF8, UCS2, UTF16) is identified in the IRM_ES field of the fixed portion of the IRM.
IRM_F2	1 byte	33	X'21'	Specifies the commit mode. • X'40' - commit mode '0' (CM0), also known as commit-then-send • X'20' - commit mode '1' (CM1), also known as send-then-commit • X'02' - specifies the SENDALTP message level activation for a commit-then-send send-receive transaction. • X'01' - specifies to IMS Connect to generate a unique client_ID to prevent a duplicate client_ID error condition. If this value is not supplied from the client, the user exit must use a default value. The commit mode flag is returned to IMS Connect from the exit in the OTMA header field, OMHDRSYN.

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_F3	1 byte	34	X'22'	Depending on the value, this field specifies the sync level, the purge or reroute option for CM0 output, or the serial delivery option for send-only input. • X'00' - sync level is 'NONE' • X'01' - sync level is 'CONFIRM' • X'02' - sync level is 'SYNCPT' • X'04' - purge undeliverable CM0 output (IRM_F3_PURGE) • X'08' - reroute undeliverable CM0 output (IRM_F3_REROUT) • X'10' - send-only with serial delivery (IRM_F3_ORDER). IRM_F3_ORDER invokes the serial (ordered) delivery option for a send-only transaction • X'20' - for multi-segment CM0 output messages, ignore the DL/I PURG call to the TP PCB (IRM_F3_IPURG) until all segments of the output message have been inserted to the TP PCB. This option is useful only if the IMS application issues a PURG call for each segment of a multi-segment message that it inserts to the TP PCB. • X'40' - for OTMA CM0 input messages, regardless of the transaction response mode, to receive a DFS2082 message (IRM_F3_DFS2082) if the IMS application does not reply to the IOPCB or complete a message switch to another transaction. This DFS2082 message for a CM0 input message applies only to the original input transaction and does not support a program-to-program switch. • X'80' - cancel duplicate client ID (IRM_F3_CANCID). On the first request of a new session, if an existing session is using the same client ID as the new session, IRM_F3_CANCID terminates the existing session to allow the new session to proceed. For CM0, the sync level must be set to confirm. If the synch level is not supplied from the client, the user exit must use a default value. The sync level flag is returned to IMS Connect from the exit in the OTMA header field, OMHDRSLV. The purge not deliverable flag is returned to IMS Connect from the user message exit in the OTMA header field, OMHDRCFL, with the setting of OMHDRPND X'10'. If the reroute flag is set, the IRM_REROUT_NM field is optional. You cannot specify the purge not deliverable function and the reroute function at the same time. If both functions are specified, the output messages are not purged or rerouted and OTMA issues message DFS2407W.

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_F4	1 byte	35	X'23'	The IRM_F4 flag identifies the type of message being sent by the client. Message types are specified by an ASCII or EBCDIC character value. The value is sent to IMS Connect , passed to the user exit, the exit builds the appropriate OTMA structure and returns it to IMS Connect to be forwarded to IMS. The valid values and the message types that they indicate are as follows: **A** ACK (IRM_F4_ACK) - An ACK response to output received from IMS Connect. ACK is used by a client to indicate the acceptance of either: • A synchronous callout request message issued by an IMS application program • An output message when the original input message from the client specifies a SYNC level of CONFIRM **C** Cancel IRM timer (IRM_F4_CANTIMER) - A request to cancel the IRM timer for another connection on which the client, using the same client ID, is waiting for output data. **D** Deallocate (IRM_F4_DEALLOC) - A request to deallocate the conversation. **J** Send-only with error (IRM_F4_SNDONLYE) - A send-only error message that returns any IMS Connect error back to the client. Errors that occur in IMS will not be returned to the client. The SNDONLYE interaction must use CM0. **K** Send only requires ACK (IRM_F4_SNDONLYA) - A send-only transaction message that requires an ACK response from IMS Connect. **L** Synchronous callout response message requiring ACK (IRM_F4_SYNRESPA) - A synchronous callout response message that uses the send-only protocol and requires an acknowledgement from IMS Connect. **M** Synchronous callout response message (IRM_F4_SYNRESP) - A synchronous callout response message that uses the send-only protocol. **N** NAK (IRM_F4_NACK) - A NAK response from the client that indicates the rejection of one of the following types of output from IMS Connect : • A synchronous callout request message issued by an IMS application program • An output message when the original input message from the client specifies a SYNC level of CONFIRM **R** RESUME TPIPE (IRM_F4_RESUMET) - A RESUME TPIPE call for asynchronous output data from IMS. A RESUME TPIPE call must execute on a transaction or persistent socket using CM0.

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_F4 (cont'd)	1 byte	35	X'23'	**S** Send only (IRM_F4_SENDONLY) - A send only transaction message that starts a send-only interaction for a non-response mode, non-conversational transaction. If the host application terminates without issuing an ISRT to the IO PCB, no DFS2082 messages are returned to the client. The SENDONLY interaction must use CM0. **blank (X'40')** A send-receive interaction for a conversational or non-conversational response mode transaction.
IRM_TRNCOD	8 bytes	36	X'24'	Character string. It specifies the IMS transaction code.
IRM_IMSDESTID	8 bytes	44	X'2C'	Character string. It specifies the data store name (IMS destination ID). This field must be specified by the client. The data store name is returned to IMS Connect from the exit in the OTMA header field, OMUSR_DESTID.
IRM_LTERM	8 bytes	52	X'34'	Character string. It specifies the IMS logical terminal (LTERM) override name. This field can be set to a valid name or to blanks. The LTERM override name is returned to IMS Connect from the exit in the OTMA header field, OMHDRLM. For IMS host applications, the value for this field is set by the user message exit, which either moves this value to the OTMA field OMHDRLTM or sets OMHDRLTM with a predetermined value. If you have specified an LTERM override value, OTMA places that value in the IOPCB LTERM field. If you do not specify an LTERM override value, OTMA instead places the IMS Connect-defined tpipe name in the IOPCB LTERM field. The tpipe name is set to the CLIENT ID if the commit mode is zero; it is set to the PORT ID if the commit mode is one. If you use the LTERM value in the IOPCB to make logic decisions, be aware of the naming conventions of the IOPCB LTERM name. If you need to use the LTERM override name for calling IBM Workload Manager for input transactions, you must specify **WLMLTRM=**YES in the DFSOTMA descriptor.
IRM_RACF_USERID	8 bytes	60	X'3C'	Character string. It specifies the RACF user ID. The client must provide it if RACF is to be used. The RACF user ID name is returned to IMS Connect from the exit in the OTMA header field, OMSECUID.
IRM_RACF_GRNAME	8 bytes	68	X'44'	Character string. It specifies the RACF group name. The RACF group name is returned to IMS Connect from the exit in the OTMA header field, OMSECGRP. To preserve the expected offset of this field, when you specify IRM_RACF_GRNAME, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_RACF_PW	8 bytes	76	X'4C'	Character string. It specifies the RACF PassTicket or password. The client must provide it if RACF is to be used. The PassTicket or password value is returned to IMS Connect from the user message exit, in the OTMA header field, OMUSR_PASSTICK. To preserve the expected offset of this field, when you specify IRM_RACF_PW, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID • IRM_RACF_GRNAME The RACF password can contain any of the following special characters. IMS Connect uses EBCDIC page 037 to validate password characters. The symbols shown are for EBCDIC code page 1047 and 037. • . (Hex value is 4B) • < (Hex value is 4C) • + (Hex value is 4E) • | (Hex value is 4F) • & (Hex value is 50) • ! (Hex value is 5A) • * (Hex value is 5C) • - (Hex value is 60) • % (Hex value is 6C) • _ (Hex value is 6D) • > (Hex value is 6E) • ? (Hex value is 6F) • : (Hex value is 7A) • = (Hex value is 7E) The RACF password can also contain the following national characters: • $ (Hex value is 5B) • # (Hex value is 7B) • @ (Hex value is 7C)
IRM_APPL_NM	8 bytes	84	X'54'	Character string. It specifies the RACF APPL name, that was defined to RACF on the PTKTDATA definition. If this field is not included or is blank, the RACF APPL name is taken from the APPL parameter of the DATASTORE configuration statement. To preserve the expected offset of this field, when you specify IRM_APPL_NM, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID • IRM_RACF_GRNAME • IRM_RACF_PW

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_REROUT_NM	8 bytes	92	X'5C'	Character string (A through Z, 0 - 9, or special characters, such as @, #, $). The reroute tpipe name of the client reroute request. This field is optional. **Recommendation:** Use blanks for the default value. When you specify the IRM_REROUT_NM field, you must also specify: • IRM_F3_REROUT in the IRM_F3 field in the user portion of the IRM • IRM_ARCH1 or IRM_ARCH2 in the IRM_ARCH field in the common fixed portion of the IRM To preserve the expected offset of this field, when you specify IRM_REROUT_NM, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID • IRM_RACF_GRNAME • IRM_RACF_PW • IRM_APPL_NM IRM_REROUT_NM and IRM_RT_ALTCID use the same offset. Both cannot be specified on the same message.
IRM_RT_ALTCID	8 bytes	92	X'5C'	Alternate client ID for a RESUME TPIPE call. This is an optional field. If the alternate client ID is provided, the IRM_ARCH field must be set to IRM_ARCH1. To preserve the expected offset of this field, when you specify IRM_RT_ALTCID, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID • IRM_RACF_GRNAME • IRM_RACF_PW • IRM_APPL_NM
IRM_TAG_ADAPT	8 bytes	100	X'64'	Name of the adapter that IMS Connect calls to convert XML messages to and from select programming languages. To preserve the expected offset of this field, when you include the IRM_TAG_ADAPT field, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID • IRM_RACF_GRNAME • IRM_RACF_PW • IRM_APPL_NM • IRM_REROUT_NM or IRM_RT_ALTCID
IRM_TAG_MAP	8 bytes	108	X'6C'	Name of the converter the XML adapter calls to perform that actual conversion of XML messages to and from select programming languages. To preserve the expected offset of this field, when you include the IRM_TAG_MAP field, you must also specify the following fields with valid values or blanks: • IRM_RACF_USERID • IRM_RACF_GRNAME • IRM_RACF_PW • IRM_APPL_NM • IRM_REROUT_NM or IRM_RT_ALTCID • IRM_TAG_ADAPT
IRM_MODNAME	8	116	74	The MFS MOD name for input messages.

Table 18. User portion of IRM for HWSSMPL0, HWSSMPL1, and user-written user message exits (continued)

Field	Length	Offsets		Meaning
		Dec	Hex	
IRM_CORTKN	0XL40			The correlation token for a synchronous callout message. The correlation token includes information that correlates a reply to a synchronous callout request with the IMS system and IMS application program that issued the request.
IRM_CT_LEN	2	124	7C	The length of the correlation token.
IRM_CT_RESV1	2	126	7E	Reserved.
IRM_CT_IMSID	4	128	80	The IMSID of the IMS system that scheduled the IMS application that issued the synchronous callout request.
IRM_CT_MEMTK	8	132	84	The OTMA TMEMBER token.
IRM_CT_AWETK	8	140	8C	OTMA message token.
IRM_CT_TPIPE	8	148	94	OTMA TPIPE name.
IRM_CT_USERID	8	156	9C	The user ID specified in the ICAL call.
IRM_SESTKN	8	164	A4	Session token value used for IMS Connect-to-IMS Connect connections.
IRM_EXTN_OFF	2	172	AC	Offset value from the start of the IRM to the first IRM extension.
IRM_F6	1	174	AE	The purpose of this flag depends on its value. **X'80' - IRM_F6_NWSE** Valid for RESUME TPIPE requests. The user-written client application supports output messages from IMS that include network security segments containing distributed network security credentials. Network security segments include the *NETSID* segment, which contains a network session ID, and the *NETUID* segment, which contains a network user ID.
IRM_RESERVED1	1	175	AF	Reserved.
		ORG IRM_MAP		
IRM_XID	XL140	32	20	The IRM_XID field is used for two-phase commit protocol messages. The IRM_XID field is only valid for IMS TMRA messages with IRM_ARCH less than 5. For IRM_ARCH = 5 and greater, you must use the value in the IRM_LEN field to determine the location of the XID in the input buffer. This must be done because the IRMMASK has been extended to support variable length IRM extensions. The IRM extensions are included as part of the IRMMASK and they contribute to the total length specified in IRM_LEN. Thus, the XID will start at the byte following the end of the now larger IRMMASK, which might also contain IRM extensions. For IRM_ARCH = 5 and greater, the message might look as follows: `LLLL | IRM + IRM EXTENSION(S)| XID | OTMA HEADERS...`
		ORG IRM_MAP: Remapping of offsets 32-35.		
IRM_TMRA_EXTN_OFF	2	32	20	Offset value from the start of the IRM to the first IRM extension in IMS TMRA messages with IRM architecture level, IRM_ARCH, equal to or greater than 5.
IRM_TMRA_SRVD1	2	34	22	Reserved for IMS Connect.
IRM_TMRA_EXTN_START	X	36	24	Denotes the start of the first IRM extension for IMS TMRA messages.
*IRM_XID_ARCH5	XL140	X	X	This field represents the size of the XID for IMS TMRA messages that use IRM_ARCH equal to or greater than 5. The location of this field in memory varies depending on the number of IRM extensions and their sizes. Therefore, the value in IRM_LEN must be used to determine the actual location of the XID. For more information, see the Meaning column for IRM_XID.

Format of IRM extensions

You can use IMS request message (IRM) extensions to send information from IMS Connect client applications without expanding the DSECT that maps the IRM.

The number of IRM extensions in each IRM can vary, and you can define the IRM extensions in the IRM in any order. All IRM extensions contain a 12-byte header. User message exits use the unique identifier (ID) for each IRM extension to determine the contents of the extension. By default, the user message exits process IRM extension IDs as EBCDIC and IRM extension data as binary data. If the IRM extension ID and extension data are not defined in EBCDIC and binary data respectively, modify the user message exit or the TCP/IP application.

Different IRM extensions are supported by different user message exits. If an IRM containing IRM extensions is processed by a user message exit that does not support the extensions, the extensions are ignored.

Table 19. Format of IRM extensions

Field	Length	Hexadecimal offset	Description and settings
IRMEXTN_LL	2	0	Length of this IRM extension.
IRMEXTN_ZZ	2	2	Reserved field.

Table 19. Format of IRM extensions (continued)

Field	Length	Hexadecimal offset	Description and settings
IRMEXTN_ID	8	4	The unique ID for this IRM extension. Each ID begins and ends with an asterisk (*). The following IDs can be defined for this field: ***CONTXT*** Indicates that the IRM extension contains context information. This ID is encoded in EBCDIC or ASCII and is supported by the HWSSOAP1 user message exit. ***TRCKID*** Indicates that the IRM extension contains a tracking ID. This ID is encoded in EBCDIC or ASCII and is supported by the HWSSOAP1 user message exit. ***PHRASE*** Indicates that the IRM extension contains a RACF password phrase. This ID is encoded in EBCDIC and is supported by the HWSJAVA0 user message exit. ***NETUID*** Indicates that the IRM extension contains a network security user ID. This ID is encoded in EBCDIC or ASCII and is supported by the HWSSMPL0 and HWSSMPL1 user message exits. ***NETSID*** Indicates that the IRM extension contains a network security session ID. This ID is encoded in EBCDIC or ASCII and is supported by the HWSSMPL0 and HWSSMPL1 user message exits.
IRMEXTN_DATA	*	12	This field can be used to locate the variable length data that immediately follow the IRM extension ID. The structure of this data depends on the IRM extension ID. By default, the IRM extension data is treated as binary data.

Output structure from client exit

The following table shows the structure (one occurrence per message) of the message returned by the user-written client application exit routine. The table lists the field name, the length of the field, and a brief explanation of the field.

Table 20. Message structure returned by user-written IMS Connect client application exit routine

Field	Length	Meaning
BPE header	64 bytes	Defined in "BPE header format" on page 230.
OTMA structure	Total length of OTMA header	For more information about the HWSOMPFX macro (full OTMA structure) see, "Macros that support IMS Connect exit routines" on page 162.
LLZZTRANCODEDATA	n bytes	• LL - length of segment • ZZ - set to binary zeros • TRANCODE - IMS 1-8 byte transaction code • DATA - user data
LLZZDATA The LLZZDATA is repeated to a maximum of 32 KB overall length. If there is more data, then the structures continues as shown in "Other IMS Connect structures" on page 229. Only the first segment will contain the IMS transaction code and the following segment will contain the segment data necessary for the transaction to process.	n bytes	• LL - length of segment • ZZ - set to binary zeros • DATA - user data
LL	2 bytes	LL - set to binary zeros to denote the end of this structure. The LL field is not part of the segment length.

Related reference

"OTMA header fields used by IMS Connect" on page 251
IMS Connect uses fields in the headers of messages sent to OTMA to communicate processing options and other information to IMS.

Other IMS Connect structures

In addition to the IRM, messages handled by IMS Connect contain other recurring structures.

The following table shows other structures that are repeated until all data has been mapped to be returned to IMS Connect. The table lists the field name, the length of the field, and a brief explanation of the field.

Table 21. Other repeated IMS Connect message structures

Field	Length	Meaning
BPE header	64 bytes	Defined in "BPE header format" on page 230.
OTMA structure	32 bytes	The OTMA message control structure that is mapped by the IMS Connect macro HWSOMPFX.
LLZZDATA The LLZZDATA field is repeated to a maximum of 32 KB overall length. If there is more data, then the structures continue.	n bytes	• LL - length of segment • ZZ - set to binary zeros • DATA - user data
LL	2 bytes	LL - set to binary zeros to denote the end of this structure.

Related reference

"OTMA message-control fields used by IMS Connect" on page 251
The table in this topic defines the fields of the OTMA message control header and the order of those fields.

BPE header format

The BPE header is a recurring structure in messages that are handled by IMS Connect.

The following table lists the fields in the BPE header layout. **This topic contains Product-sensitive Programming Interface information.**

Table 22. BPE header layout

Field	Length	Meaning
llll	4 bytes	The length of the total structure and it is set for the first BPE header only. This field is managed by IMS Connect and must not be altered by the exit.
CHAIN PTR	4 bytes	The chain pointer to the next BPE header within this message. The last BPE header in the message must have binary zeros as a chain pointer value to denote the end of the BPE headers within the message. These chain pointers are set by the user-written client application exit routine.
STORAGE TYPE	8 bytes	This field is managed by IMS Connect and should not be modified by the user exit.
TYPE ACCESS	4 bytes	This field is managed by IMS Connect and should not be modified by the user exit.

Table 22. BPE header layout (continued)

Field	Length	Meaning
SUBPOOL	1 byte	This field is managed by IMS Connect and should not be modified by the user exit.
Reserved	43 bytes	This field is managed by IMS Connect and should not be modified by the user exit.

Message structures and IMS Connect user message exit routines

IMS Connect allows up to 254 user exits to be defined in the configuration file. There are two input message structures supported by IMS Connect and two message structures supported on return from a user exit.

Input messages from client

The structure of input messages passed to and returned from a user message exit routine differs based on a number of factors, including whether the user message exit routine supports the IMS TM Resource Adapter or a user-written IMS Connect client.

The following table shows the structure for input messages received from the client by IMS Connect. The table provides information about the input message structure type, if the OTMA header is present or not, if the exit data is translated by the client code, the exit type flag, and the supporting message type.

Table 23. Input message structure

Input message structure type	OTMA header present	Exit data translated by client code	Exit type flag (IRMHDR_FLG5)	Supporting message type
1	Y	Y	11000000	HWSJAVA0
1	Y	N	10000000	HWSSMPL0 and HWSSMPL1 modified not to build OTMA headers when the client/server builds OTMA headers
2	N	Y	01000000	HWSSMPL0 and HWSSMPL1 modified not to translate data
2	N	N	00000000	HWSSMPL0 HWSSMPL1 HWSCSLO0

The following table shows the structure for input messages returned by the exit based on the input structure received by the exit. The table provides information about the input message structure type, the exit output message structure type, the exit type flag, and the supporting message type.

Table 24. Input message structure returned by the exit

Input message structure type	Exit output message structure type	Exit type flag (IRMHDR_FLG5)	Supporting message type
1	1	11000000	HWSJAVA0

Table 24. Input message structure returned by the exit (continued)

Input message structure type	Exit output message structure type	Exit type flag (IRMHDR_FLG5)	Supporting message type
1	1	10000000	HWSSMPL0 and HWSSMPL1 modified not to build OTMA headers when the client/server builds OTMA headers
2	3	01000000	HWSSMPL0 and HWSSMPL1 modified not to translate data
2	3	00000000	HWSSMPL0 and HWSSMPL1

Output message to client

The output message from IMS is passed to the user message exit routine that was called from the client.

The user exit normally removes the OTMA headers for output if the exit added the OTMA headers for input. The user exit normally translates the data from EBCDIC to ASCII if it did the translation for input. And the reverse is true if these things were not done for input.

The OTMA header can consist of up to four sections and application data. If the exit is to remove the OTMA header (not present on input), there must be a check for each section. The four sections include:

- Control (always present in the OTMA structure)
- Header (might or might not exist in the OTMA structure)
- Security (might or might not exist in the OTMA structure)
- User (might or might not exist in the OTMA structure)

Output message from IMS-to-IMS Connect

All output messages received by IMS Connect from IMS consist of the same structure, the OTMA header followed by LLZZ DATA. If the message contains multiple segments, then the OTMA header and LLZZ DATA are repeated for the number of segments in the message.

For CM0 multi-segment output messages, if the IMS application program issues a DL/I PURG call after it inserts each segment to the TP PCB, each segment of the output message is sent to the IMS Connect client separately.

If your IMS application issues a PURG call for each segment of multi-segment CM0 output messages, you can instruct OTMA to ignore PURG calls until the last segment of the message is inserted to the TP PCB by specifying X'02' (OMHDRIPG) in the OMHDRCFL field of the state data section of the OTMA message header. When OMHDRIPG is specified, OTMA returns the output message to IMS Connect as a single multi-segment message.

User-written IMS Connect clients can set the OMHDRIPG flag by specifying X'20' (IRM_F3_IPURG) in the IRM_F3 field of the user portion of the IRM prefix of input transaction messages.

If an input message to IMS contains distributed network security credentials in the security-data section of the OTMA message prefix, output messages to IMS Connect clients that issued a RESUME TPIPE call can also contain the network security credentials. You can enable network security credentials to be included in output messages by specifying X'80' in the IRM_F6 field of the user portion of the IRM prefix of input transaction messages. If X'80' is not specified in the IRM_F6 field, IMS removes the network security credentials from the security section of the OTMA header in the output message, even if the security credentials are included in the input message.

Output message from IMS returned by the exit back to IMS Connect

The message returned to IMS Connect from the exit consists of one of two structures:

- Messages with OTMA structures imbedded in the message
- Message with no OTMA structures imbedded in the message

Message structures

The required structure of messages that flow through IMS Connect differs depending on whether the message is an input message or an output message, and whether the message has been modified by the user message exit yet. The following topics describe the message structures for the IMS TM Resource Adapter and user-written IMS Connect client applications.

Input message from client and passed to message exit

Input messages from the client consist of the IMS TM Resource Adapter and user-written IMS Connect client application message structures.

This topic contains Product-sensitive Programming Interface information.

IMS TM Resource Adapter message structure - type 1

The following table shows the input message format supported by IMS Connect from an IMS TM Resource Adapter client.

Table 25. IMS Connect message format for IMS TM Resource Adapter client applications

Field	Length	Meaning
llll	4 bytes	Length of entire message, including llll field.
IRM	28 bytes	The common, fixed portion of the IRM.
IRM_MAP	0 bytes	Position of the XID map when it is present.
OTMA HDRs	466 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
LL	2 bytes	Length of data segment.
zz	2 bytes	Reserved (set to binary zeros).
DATA	n bytes	User data with the transaction code first.
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
LL	2 bytes	Length of second data segment.
zz	2 bytes	Reserved (set to binary zeros).
DATA	n bytes	User data in the second data segment (no transaction code).
...		
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
LL	2 bytes	Length of this and last data segment.
zz	2 bytes	Reserved (set to binary zeros).

Table 25. IMS Connect message format for IMS TM Resource Adapter client applications (continued)

Field	Length	Meaning
DATA	n bytes	User data with this data segment (no transaction code).
	ORG IRM_MAP	
IRM_XID	140 bytes	Global transaction ID for two-phase commit processing.

User-written message structure - type 2

The following table shows the input message format supported by IMS Connect for user-written client applications.

The message format is supported by the HWSSMPL0 and HWSSMPL1 user message exit routines, as well as user-written user message exit routines. This message format is not supported by HWSJAVA0, HWSSOAP1, HWSCSLO0, or HWSCSLO1.

If you are using a user-written message exit routine, you can structure the message fields as required by the user exit. You can include fields that are used only by the user message exit or other fields that can be passed in the OTMA headers, such as the MID name.

The following is the data structure for all user-written client applications.

The transaction data in input messages from user-written IMS Connect client applications follows the user-defined IRM, or IRM extensions if extensions are defined, and is contained in one or more data segments. The data segments must conform to the format of LLZZDATA, where LL contains the total length of the data segment, ZZ contains binary zeros, and DATA contains the transaction code and transaction data or an IMS command.

The transaction data in input messages from user-written IMS Connect client applications follows the user-defined IRM and is contained in one or more data segments. The data segments must conform to the format of LLZZDATA, where LL contains the total length of the data segment, ZZ contains binary zeros, and DATA contains the transaction code and transaction data or an IMS command.

IMS command input must be submitted in a single data segment followed by EOM.

Table 26. IMS Connect message format for user-written client applications

Field	Length	Meaning
llll	4 bytes	Length of entire message, including llll field.
Fixed IRM prefix	28 bytes	The common, fixed portion of the IRM mapped by the HWSIMSCB macro in the IMS.SDFSMAC data set. For a description of the fields see “Format of fixed portion of IRM in messages sent to IMS Connect” on page 214.
User-defined IRM	Variable length	User-defined part of the IRM mapped by the HWSIMSCB macro in the IMS.SDFSMAC data set. For a description of the fields see “Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines” on page 218. HWSJAVA0, HWSSOAP1, HWSCSLO0, and HWSCSLO1 do not support the user-defined part of the IRM.
IRM extension	Variable length	The portion of the IRM that allows information to be sent to IMS Connect without requiring the IRM DSECT to expand. For a description of the fields see “Format of IRM extensions” on page 227.
LL	2 bytes	Length of first data segment.
zz	2 bytes	Reserved (set to binary zeros).

Table 26. IMS Connect message format for user-written client applications (continued)

Field	Length	Meaning
DATA	n bytes	User data with the transaction code first.
LL	2 bytes	Length of second data segment.
zz	2 bytes	Reserved (set to binary zeros).
DATA	n bytes	User data second data segment (no transaction code).
...		
LL	2 bytes	Length of this data segment.
zz	2 bytes	Reserved (set to binary zeros).
DATA	n bytes	User data segment (no transaction code).
LL	2 bytes	End of message (set to binary 0000 0000 0000 0100).
zz	2 bytes	Reserved (set to binary zeros).

Related reference

"Format of fixed portion of IRM in messages sent to IMS Connect" on page 214
The IMS request message (IRM) header contains a 28-byte fixed-format section that is common to all messages from all IMS Connect client applications that communicate with IMS TM.

Input message returned from message exit

Input messages from the message exit consist of the IMS TM Resource Adapter and user-written IMS Connect client application message structures.

IMS TM Resource Adapter message structure - type 1

The IMS TM Resource Adapter exit output message format that is supported by IMS Connect is the same message format of the input message.

The total length of the message can be 10,000,000 bytes. The length of each segment (from the BPE header to the next BPE header) within the message can be a maximum of 32 KB, excluding the BPE and OTMA headers.

User-written IMS Connect client applications message structure - type 3

The following table shows the output message format supported by IMS Connect from the supplied HWSSMPL0 exit (user-written IMS Connect client application exit routine). The table provides information about the field, length, and meaning. Variable length OTMA headers are supported, and therefore, the OTMA header length can be other than 466 bytes. The following example contains 466 bytes as used by the supplied exits.

Table 27. Supported output message format for the HWSSMPL0 exit

Field	Length	Meaning
BPE HEADER	64 bytes	For the format of the BPE header, see "BPE header format" on page 237.

Table 27. Supported output message format for the HWSSMPL0 exit (continued)

Field	Length	Meaning
OTMA HDRs	466 - 970 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields, see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251. If the security data section of the OTMA header contains network security information, the size of the header can be up to 504 bytes larger than without the network security information.
LL	2 bytes	Length of first data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data with the transaction code first
Repeat of ll,zz,DATA		
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data (no transaction code)
yy	2 bytes	Binary value of zero
BPE HEADER	64 bytes	For the format of the BPE header, see “BPE header format” on page 237.
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data segment (no transaction code)
Repeat of LL,zz,DATA		
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data (no transaction code)
...		
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data (no transaction code)
yy	2 bytes	Binary value of zero

Table 27. Supported output message format for the HWSSMPL0 exit (continued)

Field	Length	Meaning
BPE HEADER	64 bytes	For the format of the BPE header, see "BPE header format" on page 237.
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, "OTMA header fields used by IMS Connect," on page 251.
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data (no transaction code)
...		
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data (no transaction code)
yy	2 bytes	Binary value of zero

Restriction: The length of data from one BPE header to the next BPE header cannot exceed 32K, excluding the BPE header and the OTMA header.

BPE header format

The following table describes length and meaning of the fields in the BPE header format.

Restriction: User message exit routines must not modify any fields in the BPE header except the chain pointer field. User message exit routines modify the chain pointer field to chain the BPE headers together with the last BPE chain pointer set to binary zeros.

This topic contains Product-sensitive Programming Interface information.

Table 28. BPE header format

Field	Length	Meaning
llll	4 bytes	Length of field of entire buffer
CHAIN PTR	4 bytes	Chain pointer to next BPE header
STORAGE TYPE	8 bytes	Storage type
TYPE ACCESS	4 bytes	Type access
SUBPOOL	1 byte	Subpool
RESV	43 bytes	Reserved

Related reference

"Input message from client and passed to message exit" on page 233

Input messages from the client consist of the IMS TM Resource Adapter and user-written IMS Connect client application message structures.

Output message passed to message exit

Output messages from IMS Connect are passed to the user message exit in a certain message structure, regardless of whether the exit is for an IMS TM Resource Adapter client or a user-written IMS Connect client application.

The following table shows the message format from IMS Connect to the exit for the client. The table provides information about the length and meaning of the fields in the output message.

Table 29. Format of output messages from IMS Connect to the user message exit

Field	Length	Meaning
OTMA HDRs	Length of total OTMA headers	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
Control data (applicable to synchronous callout request)	n bytes	Control data sent by the IMS application program
LL	2 bytes	Length of data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
LL	2 bytes	Length of this segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
...		
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, “OTMA header fields used by IMS Connect,” on page 251.
LL	2 bytes	Length of this segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data

Related reference

IMS TM Resource Adapter user message exit routine (HWSJAVA0) (Exit Routines)

Output message from message exit to client

Depending on the type of IMS Connect client and the user message exit used to support the client, the format of the message structure differs.

This topic contains Product-sensitive Programming Interface information.

The output message from the message exit that is sent to IMS Connect client applications is in one of the following formats:

- "IMS TM Resource Adapter message structure" on page 239
- "User-written application message structure" on page 240

IMS TM Resource Adapter message structure

The following table shows the message format of the output message from the user message exit HWSJAVA0. The table provides information about the length and meaning of the fields in the output message.

Table 30. Output message format from the user message exit HWSJAVA0

Field	Length	Meaning
LLLL	4 bytes	Total message length
Id	8 bytes	*HWSJAV*
OTMA HDRs	466 - 970 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, "OTMA header fields used by IMS Connect," on page 251. If the security section of the OTMA header contains network security information, the size of the header can be up to 504 bytes larger than without the information.
Control data (applicable to synchronous callout request)	n bytes	Control data sent by the IMS application program
LL	2 bytes	Length of data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, "OTMA header fields used by IMS Connect," on page 251.
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
...		

Table 30. Output message format from the user message exit HWSJAVA0 (continued)

Field	Length	Meaning
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, "OTMA header fields used by IMS Connect," on page 251.
LL	2 bytes	Length of this segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
OTMA CTL HDR	32 bytes	Mapped by the HWSOMPFX macro in the IMS.SDFSMAC data set. For a description of the fields see Chapter 17, "OTMA header fields used by IMS Connect," on page 251.
LL	2 bytes	Length of data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
...		
LL	2 bytes	Length of this data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data

User-written application message structure

The user-written IMS Connect client application message structure can consist of one or more TCP/IP message structures.

Message structures returned to the client by the sample user message exit routine HWSSMPL1 (*SAMPL1*) start with LLLL, a 4-byte field that contains the total length of the message. The sample user message exit routine HWSSMPL0 (*SAMPLE*) does not add the LLLL field at the beginning of the message structures that it returns.

The output message from the message exit that is sent to user-written IMS Connect client applications is in one of the following formats:

Message format when the MFS MOD name is requested

The following table shows the format of the output message sent to the client application when the MFS MOD name is requested. The table provides information about the length and meaning of the fields in the output message.

Table 31. Output message format containing RMM, DATA, and CSM

Field	Length	Meaning
LLLL (HWSSMPL1 only)	4 bytes	The total length of an output message returned by HWSSMPL1; output messages returned by HWSSMPL0 do not include this field

Table 31. Output message format containing RMM, DATA, and CSM (continued)

Field	Length	Meaning
RMM header (optional)	20 bytes	The first four bytes of the RMM are LLzz, the two byte length of the RMM and two bytes of binary zeros
CORTKN header (applicable to synchronous callout request)	52 bytes	Correlation token for this synchronous callout request
Control data (applicable to synchronous callout request)	n bytes	Control data sent by the IMS application program
LL	2 bytes	Length of data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
LL	2 bytes	Length of second data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data second data segment
...		
LL	2 bytes	Length of nth segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data nth data segment
CSM	12 bytes	Complete status message

Message format when the MFS MOD name is not requested

The following table shows the format of the output message sent to the client application when the MFS MOD name is not requested and only data and the CSM are being sent. The table provides information about the length and meaning of the fields in the output message.

Table 32. Output Message format containing output data and CSM only

Field	Length	Meaning
LLLL (HWSSMPL1 only)	4 bytes	The total length of an output message returned by HWSSMPL1; output messages returned by HWSSMPL0 do not include this field
CORTKN header (applicable to synchronous callout request)	52 bytes	Correlation token for this synchronous callout request
Control data (applicable to synchronous callout request)	n bytes	Control data sent by the IMS application program
LL	2 bytes	Length of data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data
LL	2 bytes	Length of second data segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data second data segment

Table 32. Output Message format containing output data and CSM only (continued)

Field	Length	Meaning
...		
LL	2 bytes	Length of nth segment
zz	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data nth data segment
CSM	12 bytes	Complete status message

Format of an IMS command response message

The output message sent to a client submitting commands to the Operations Manager (OM) component of the IMS Common Service Layer (CSL) is shown in the following table. The table provides information about the length and meaning of the fields in the output message.

Table 33. Output message format sent to OM command client

Field	Length	Meaning
LLLL	4 bytes	Length of output
DATA	n bytes	IMS command output

Format of the request mod message

The IMS Connect request mod message (RMM) is returned as the first structure of an output message from the message exit if the MFS MOD name is requested and the data output is present.

The following table shows the output message format of the request mod message built by the user message exits HWSSMPL0 and HWSSMPL1. The table includes the field name, field length, and field meaning.

Table 34. Request mod message output message format

Field	Length	Meaning
LL	2 bytes	Length of RMM message
zz	2 bytes	Reserved (set to binary zeros)
ID	8 bytes	Character value of *REQMOD*
MOD	8 bytes	Character value of the requested MFS MOD name

Format of the returned client ID message

The IMS Connect returned client ID message (GENCID) is returned as the first structure (if no RMM segment) or as the second structure (follows RMM segment) of an output message from the message exit if the client ID is requested and output is present.

The following table shows the output message format of the requested client ID built by the user message exits HWSSMPL0, HWSSMPL1 and HWSDPWR1. The table includes the field name, field length, and field meaning.

Table 35. Request client ID message format

Field	Length	Meaning
LL	2 bytes	Length of GENCID message
ZZ	2 bytes	Reserved (set to binary zeros)

Table 35. Request client ID message format (continued)

Field	Length	Meaning
ID	8 bytes	Character value of (*GENCID*)
CLIENT ID	8 bytes	Character value of Client ID name

Format of the complete status message

The IMS Connect complete status message (CSM) is returned as the last structure of an output message from the message exit if the input message is processed successfully.

The following table shows the output message format of the CSM. The table includes the field name, field length, and field meaning.

Table 36. Complete status message output message format

Field	Length	Meaning
CSM_LEN	2 bytes	Length of CSM message
CSM_FLG1	1 byte	Flag byte: **X'80'** Asynchronous message queued in IMS **X'40'** Conversational output message **X'20'** ACK/NAK required **X'10'** Protocol level available
CSM_PRLVLFLG	1 byte	IMS Connect protocol level flag: **X'00'** Base protocol level. **X'01'** Reserved. **X'02'** Support for CM0 ACK NOWAIT transactions is enabled.
CSM_ID	8 bytes	Character value of *CSMOKY*

Format of the request status message

The IMS Connect request status message (RSM) is returned as the only structure of an output message from the message exit if IMS Connect or the message exit determined an error occurred.

The following table shows the output message format of the request status message for an error condition. The table includes the field name, field length, field meaning.

Table 37. Request status message output message format

Field	Length	Meaning
RSM_LEN	2 bytes	Length of RSM message

Table 37. Request status message output message format (continued)

Field	Length	Meaning
RSM_FLG1	1 byte	Flag byte: **X'80'** Asynchronous message queued in IMS **X'40'** Conversational output message **X'20'** ACK/NAK required
RSM_RACFRC or RSM_OTMRSN	1 byte	If this RSM is generated by a RACF security error, this is the RSM_RACFRC field, which contains the return code from a **RACROUTE REQUEST=VERIFY** command. Otherwise, this is the RSM_OTMRSN field, which contains a reason code from OTMA.
RSM_ID	8 bytes	Character value of *REQSTS*
RSM_RETCOD	4 bytes	Return code
RSM_RSNCOD	4 bytes	Reason code

Format of the PING response

The PING response is sent to IMS Connect clients after you send `PING IMS_CONNECT`, a ping request, to IMS Connect.

The following table shows the format of the PING response data built by the user message exit HWSJAVA0. The table includes the field name, field length, and field meaning.

Table 38. PING response output data format

Field	Length	Meaning
PNG_LEN	2 bytes	Length of PING response data
PNG_FLG1	1 byte	Flag byte: **X'02'** IMS Connect supports network security credentials in the security section of the OTMA header that is sent by IMS Transaction Message Resource Adapter (IMS TM resource adapter). **X'01'** Support for IMS extensions present (Requires IRM_ARCH=5)
PNG_FLG2	1 byte	Reserved
PNG_RESP	15 bytes	Might contain '*PING RESPONSE*'

Table 38. PING response output data format (continued)

Field	Length	Meaning
ORG	PNG_RESP	Remapping of PNG_RESP
PNG_RESP1	25 bytes	Might contain 'HWSC0030I *PING RESPONSE*'

Format of the correlation token for synchronous callout messages from IMS applications

The output messages from the IMS Connect message exit for synchronous callout requests include a correlation token to correlate the reply to the callout request with the IMS application that issued the request.

The correlation token is the first structure in a synchronous callout request and can be followed by the optional request mod message (RMM) structure and the application data.

The following table shows the structure of the correlation token on output from IMS Connect, as mapped by the CORMask DSECT in the HWSIMSCB macro:

Table 39. Format of the correlation token for synchronous callout requests in output messages

Field	Length	Meaning
COR_Len	2 bytes	Length of correlation structure, including the structure header and the correlation token
COR_Rsvd	2 bytes	Reserved (set to binary zeros)
COR_Id	8 bytes	Character value of *CORTKN*
COR_LL	2 bytes	Length of the correlation token
COR_PSTNR	2 bytes	Region ID for the synchronous callout message
COR_IMSID	4 bytes	ID of the IMS system in which the IMS application program that issued the callout request is running
COR_MEMTK	8 bytes	OTMA TMEMBER token
COR_AWETK	8 bytes	OTMA message token
COR_TPIPE	8 bytes	OTMA Tpipe name
COR_USERID	8 bytes	User ID included in the ICAL call by the IMS application that issued the callout request

Related tasks

“Configuring user-written IMS Connect clients for synchronous callout requests” on page 201

To support synchronous callout requests, user-written IMS Connect clients must be configured to retrieve new callout requests from IMS, acknowledge the receipt of the callout request (ACK or NAK), and to return the synchronous callout responses to IMS through IMS Connect.

Format of the callout control data in the message

The format of the callout control data in the output message that is sent to the client application depends on the type of client application that issues the resume tpipe request.

For non-IMS TM resource adapter clients

The output message that is sent by IMS Connect to the client application includes a structure that contains control data. This structure can be mapped by using the CTLDATA and CTLDATASEG DSECTs, which are included in the macro HWSIMSCB. The control data structure is placed before the callout request data.

The format of the control data structure is as follows:

Table 40. Callout control data message format for non-IMS TM resource adapter clients

Field	Length	Meaning
CTL_LLLL	4 bytes	Length of control data structure
CTL_ID	8 bytes	Control data structure ID, which is a character value of *CTLDAT*
zzzz	4 bytes	Reserved for IMS Connect
CTL_HDRLEN1	4 bytes	Length of the header (tag) of the first control data section. This field is followed by the first pair of open and close tags.
CTL_DATA1	*d1*	The first control data section, which is in the following format:`<tag1>data1</tag1>`
CTL_HDRLEN2	4 bytes	Length of the header (tag) of the second control data section. This field is followed by the second pair of open and close tags.
CTL_DATA2	*d2*	The second control data section, which is in the following format:`<tag2>data2</tag2>`
:	:	:
CTL_HDRLEN*n*	4 bytes	Length of the header (tag) of the last control data section. This field is followed by the last pair of open and close tags.
CTL_DATA*n*	*dn*	The last control data section, which is in the following format:`<tag`*n*`>data`*n*`</tag`*n*`>`

For IMS TM resource adapter clients

The format of the control data structure is as follows:

Table 41. Callout control data message format for IMS TM resource adapter clients

Field	Length	Meaning
LLLL	4 bytes	Length of control data structure
LLLL	4 bytes	Length of the header (tag) of the first control data section. This field is followed by the first pair of open and close tags.
DATA	*d1*	The first control data section, which is in the following format: `<tag1>data1</tag1>`
LLLL	4 bytes	Length of the header (tag) of the second control data section. This field is followed by the second pair of open and close tags.
DATA	*d2*	The second control data section, which is in the following format: `<tag2>data2</tag2>`
:	:	:
LLLL	4 bytes	Length of the header (tag) of the last control data section. This field is followed by the last pair of open and close tags.
DATA	*dn*	The last control data section, which is in the following format: `<tag`*n*`>data`*n*`</tag`*n*`>`

Format of the returned network security segments

The Network Session ID (NETSID) and Network User ID (NETUID) segments contain network security information and are returned to the clients when the clients issue RESUME TPIPE calls.

The NETSID and NETUID segments are returned to clients only if both of the following situations occur:

- You specify X'80' (IRM_F6_NWSE) in the IRM_F6 field of the IRM prefix.
- IRM extensions with an ID of *NETUID*, or *NETSID*, or both are included in IMS Connect client input messages.

The following tables show the format of the output segments that contain network security information. The information is built by the HWSSMPL0 and HWSSMPL1 user message exits. The tables include the field name, field length, and field meaning.

Table 42. NETSID segment format

Field	Length in bytes	Meaning
LL	2	Length of NETSID segment
RESERVED	2	Reserved (set to binary zeros)
ID	8	Character value of (*NETSID*)
RESERVED	4	Reserved (set to binary zeros)
DATA	n	The data for this segment. The length of this data is variable.

Table 43. NETUID segment format		
Field	**Length in bytes**	**Meaning**
LL	2	Length of NETUID segment
RESERVED	2	Reserved (set to binary zeros)
ID	8	Character value of (*NETUID*)
RESERVED	4	Reserved (set to binary zeros)
DATA	n	The data for this segment. The length of this data is variable.

Examples of message structures in a simple interaction

The following examples show the message structures in a simple interaction between IMS Connect and a user-supplied client application program.

The flow represented in each example is a simple send-receive transaction that uses commit-mode 1 and sync level=confirm, as defined by the following IRM values:

- IRM_F2 = X'20' (for CM1)
- IRM_F3 = X'01' (for sync level = confirm)
- IRM_F4 = X'40' (for a send-receive interaction)

HWSSMPL0 message structures in a simple interaction

The example flow shown in the following table is a simple send-receive transaction between a user-supplied IMS application program and IMS Connect when the sample user message exit routine HWSSMPL0 is used (IRM_ID = *SAMPLE*).

In the table:

- The flow starts at the top of the table and progresses down.
- The vertical bars separate the various structures within each message.
- The input transaction and transaction response can have one or more LLZZ data structures.
- The LL RMM structure shown in the transaction response is optional.
- 0004 0000 is the 4 byte end-of-message structure.

Table 44. Example message structures for HWSSMPL0 in a simple interaction

Client state	Example message structure
Client send (input transaction)	LLLL | LL IRM | LLZZ data | LLZZ data | 0004 0000
Client receive (transaction response)	LL RMM | LLZZ data | LLZZ data | LL CSM
Client send (ACK)	LLLL | LL IRM | 0004 0000
Client receive (deallocate confirm)	LL RSM

HWSSMPL1 message structures in a simple interaction

The example flow shown in the following table is a simple send-receive transaction between a user-supplied client application program and IMS Connect when the sample user message exit routine HWSSMPL1 is used (IRM_ID = *SAMPL1*).

In the table:

- The flow starts at the top of the table and progresses down.

- The vertical bars separate the various structures within each message.
- The input transaction and transaction response can have one or more LLZZ data structures.
- The LL RMM structure shown in the transaction response is optional.
- 0004 0000 is the 4 byte end-of-message structure.

Table 45. Example message structures for HWSSMPL1 in a simple interaction

Client state	Example message structure
Client send (input transaction)	LLLL | LL IRM | LLZZ data | LLZZ data | 0004 0000
Client receive (transaction response)	LLLL | LL RMM | LLZZ data | LLZZ data | LL CSM
Client send (ACK)	LLLL | LL IRM | 0004 0000
Client receive (deallocate confirm)	LLLL | LL RSM

Chapter 17. OTMA header fields used by IMS Connect

IMS Connect uses fields in the headers of messages sent to OTMA to communicate processing options and other information to IMS.

OTMA defines the format and valid values of the message headers. The fields of the OTMA message header are grouped into sections based on the purpose of the fields.

Depending on the type of message being sent or the options you select, the format of a given section of the OTMA header can change. For example, depending on the type of message being submitted, the format of the message-control section of the OTMA header differs.

If the security-data section of the OTMA header contains the NETSID section or the NETUID section, or both, for network security credentials, you might need to modify code that uses the HWSOMPFX macro to map the header. If network security credentials are included in the security-data section, the size of the security section might vary and cause the locations of the fields that are below the security section to also change and become inaccessible. To inspect or modify any section of the OTMA header so that the data in the header can be accessed, do the following steps:

1. Inspect the values of the OMCTLPFL field to determine whether the state data, security data, user data, and application data sections exist in the message.
2. To access a specific section in the OTMA header, use the values of the OMCTLLEN, OMHDRLEN, OMSECLEN, and OMUSRLEN fields, if present, to skip over the message's control data, state data, security data, and user data sections, respectively.

The tables in the following subsections document the IMS Connect requirements for setting the OTMA header fields or integrating them with the IMS Connect user message exits. The tables correspond to the sections of the OTMA header. Additional notes for the fields listed in each table are denoted by a number in the final column. The corresponding note text is listed by number in “Notes to OTMA header tables” on page 281.

Related reference

“OTMA message prefix” on page 859
OTMA messages have a prefix that conforms to a format that is mapped by the DFSYMSG macro in the IMS.ADFSMAC data set.

OTMA message-control fields used by IMS Connect

The table in this topic defines the fields of the OTMA message control header and the order of those fields.

The numbered notes in the table are explained in “Notes to OTMA header tables” on page 281.

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLALV	1	0		ARCHITECTURE LEVEL Set to X'01' architecture level 1. Set for all messages.	1

© Copyright IBM Corp. 1974, 2020

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLMGT	1	1	MESSAGE TYPE		
			OMCTLDTA X'80'	MESSAGE TYPE=Data Set for conversational transactions but not on first input. If EXPREA FLAG1 is set to EXPREA_ CONVERS then set OMCTLMGT to OMCTLDTA. EXPREA FLAG1 is not set to EXPREA_ CONVERS on the first input for conversation.	1
			OMCTLTXN X'40'	MESSAGE TYPE=Transaction Set for first transaction input. That is, first input for conversation or nonconversation, EXPREA_FLAG1 is not set to EXPREA_CONVERS.	1
			OMCTLRSP X'20'	MESSAGE TYPE=Response Set for ACK or NAK response to message sent to client. Required for: • Commit Mode 0 (synch level=CONFIRM) • Commit Mode 1 (synch level=CONFIRM)	1
			OMCTLCMD X'10'	MESSAGE TYPE-Command Set for - RESUME TPIPE calls	1
			OMCTLCMT X'08'	MESSAGE TYPE = Commit Confirmation Set for SEND ONLY or DEALLOCATE. SEND ONLY or DEALLOCATE is indicated in IRM from client.	1

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLRSI	1	2	RESPONSE INDICATOR		
			OMCTLACK X'80'	RESPONSE = ACK Set for ACK. ACK is indicated in IRM.	
			OMCTLNAK X'40'	RESPONSE = NAK Set for NAK. NAK is indicated in IRM.	1
			OMCTLRRQ X'20'	RESPONSE = Response requested If set, then conversational transaction and the IMSEA_RSNCODE must be set to 96 (X'60') to signal client application that conversation continues.	1
			OMCTLERQ X'10'	RESPONSE=Extended response requested. Neither tested nor set by exit.	4
			OMCTLSYR X'08'	Response to a synchronous callout message	
			OMCTLDAN X'02'	Support for delayed ACK or NAK response	
OMCTLCCI	1	3	COMMIT CONFIRMATION INDICATOR AND OTHER FLAGS		
			OMCTLCTD X'80'	Confirm=Committed If set, then the IMS application has terminated the conversation, and the IMSEA_RSNCODE must be set to 97 (X'61') to signal client application that the IMS application terminated successfully.	
			OMCTLABT X'40'	Confirm=Aborted. Neither tested nor set by exit.	4
			OMCTSATP EQU'04'	The SENDALTP message level activation set whether the IRM requests the SENDALTP support, and whether field IRM_F2 is set to "IRM_F2_SNDALTP". This is only valid for a commit-then-send send-receive transaction.	

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLTYP	1	4	COMMAND TYPE		
			OMCTLBID X'04'	COMMAND=Client Bid. Neither tested nor set by exit.	4
			OMCTLAVL X'08'	COMMAND=Server Available. Neither tested nor set by exit.	4
			OMCLTRSN X'0C'	Command=Resynch. Neither tested nor set by exit.	4
			OMCTLSPA X'14'	Command=Suspend I/P for all tpipes. Neither tested nor set by exit.	4
			OMCTLRSA X'18'	Command=Resume I/P for all tpipes. Neither tested nor set by exit.	4
			OMCTLSPN X'1C'	Command=Suspend I/P for named tpipe. Neither tested nor set by exit.	4
			OMCTLRSM X'20'	Command=Resume I/P for named tpipe. Neither tested nor set by exit.	4
			OMCTLRTP X'24'	Command=Resume O/P for named tpipe without options. Set for RESUME TPIPE call without options.	1
			OMCTLRID X'28'	Command=Resume single tpipe with options. Set for RESUME TPIPE call with options.	1
			OMCTLMTR X'3C'	Command=Resource state protocol command. OTMA sends this command to notify IMS Connect and other OTMA clients about how well the IMS system is processing the input received from OTMA clients. IMS Connect updates the exit interface block data store entry (HWSXIBDS) with the information sent in this command.	

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLPFG	1	5	PROCESSING FLAG		
			OMCTLLPG X'80'	Load Program. Neither tested nor set by exit.	4
			OMCTSHDN X'80'	The suspended processing for all tpipes (OMCTLSPA) is due to IMS shutdown.	7
			OMCTLSYP X'40'	Synchronized tpipe. Neither tested nor set by exit.	4
			OMCTLASY X'20'	Asynchronous/ unsolicited queued messages. Neither tested nor set by exit.	4
			OMCTLERR X'10'	There is an error message with the NAK. Neither tested nor set by exit.	4
			OMCTLQUE X'08'	Asynchronous message is in IMS Hold Queue. If set, set CSM_FLG1 to CSM_AMSG if sending CSM, or set RSMFLG1 to RSM_AMSG if sending RSM.	4
OMCTLOME X'01'	SCI not present error message.				
OMCTLTNM	8	6		Tpipe name. Neither tested nor set by exit.	4
OMCTLCHN	1	E	CHAIN STATE FLAG		
			OMCTLFIC X'80'	First in chain. Set for first message segment in chain.	1
			OMCTLMIC X'40'	Middle in chain. Set for not first and, or, not last message segment in chain.	1
			OMCTLLIC X'20'	Last in chain. Set for last message segment in chain.	1
			OMCTLCAN X'10'	Cancel this message. Neither tested nor set by exit.	4

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLPFL	1	F	PREFIX FLAG		
			OMCTLSTD X'80'	State Data is present. Set if State Data Header present in OTMA Headers being built.	1
			OMCTLSEC X'40'	Security data is present. Set if Security Data Header present in OTMA Headers being built.	1
			OMCTLUSR X'20'	User data is present. Set if User Data Header present in OTMA Headers being built.	1
			OMCTLAPP X'10'	Application data is present. Set if Application Data Header present in OTMA Headers being built.	1
OMCTLSSN	4	10		SEND SEQUENCE NUMBER. NEITHER TESTED NOR SET BY EXIT.	4
OMCTLSNS	4	14		SENSE CODE. See OMCTLSNC and OMCTLRSC, which follow.	
		ORG OMCTLSNS			
OMCTLSNC	2	14		SENSE CODE If nonzero value, then build a NAK RSM to send to the client application, pass the sense code in the RSM as the reason code, and set the return code to X'0C'.	1
OMCTLRSC	2	16		REASON CODE NEITHER TESTED NOR SET BY EXIT.	4
OMCTLRSQ	4	18		RECOVERABLE MESSAGE SEQUENCE NUMBER NEITHER TESTED NOR SET BY EXIT.	4

Table 46. HWSOMCTL DSECT - OTMA message-control header. Control data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMCTLSEQ	2	1C		SEGMENT SEQUENCE NUMBER Set to 1 in first OTMA Control Header and count maintained in user work area. Increment by 1 for each subsequent OTMA Control Header within a single message being sent to IMS.	1
	1	1E		RESERVED.	3
OMCTLCDN	1	1F		Specifies the number of control data segments (1 to 255) when OMHDRCTD is set in the state data. This field is used only when IMS sends an ICAL callout message with control data in the application data section of the message.	
		ORG OMCTLRSQ			
OMCTLUID	8	18		The user ID of the client that is submitting a RESUME TPIPE call	

Related reference

"Message-control information section" on page 859
For every OTMA message, you must provide message-control information in the first section of the OTMA message prefix.

OTMA state data fields used by IMS Connect

The tables in this topic describe the fields of the OTMA state data header and the order of those fields.

The numbered notes in each table are explained in "Notes to OTMA header tables" on page 281.

Server Available and Client Bid command format

Table 47. HWSOMHDR DSECT - OTMA state data header. State data common section for Server Available and Client Bid command format

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRLEN	2	0		STATE DATA LENGTH Set to State Data length.	1
OMHDRORG		2	State data for 'Server Available' and 'Client Bid' commands.		
OMHDRONM	16	2		MEMBER NAME OF ORIGINATING SERVER. NEITHER TESTED NOR SET BY EXIT.	4

Table 47. HWSOMHDR DSECT - OTMA state data header. State data common section for Server Available and Client Bid command format (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDROMT	8	12		MEMBER TOKEN OF COMMAND ORIGINATOR. NEITHER TESTED NOR SET BY EXIT.	4
OMHDRDMT	8	1A		MEMBER TOKEN OF COMMAND DESTINATION. NEITHER TESTED NOR SET BY EXIT.	4
OMHDRUEN	8	22		UNRESOLVED DESTINATION EXIT NAME. NEITHER TESTED NOR SET BY EXIT.	4
OMHDRMBS	2	2A		XCF TRANSMISSION MAX BLOCKSIZE. NEITHER TESTED NOR SET BY EXIT.	4
OMHDRQUE	1	2C	OMHDRCMQ X'80'	CREATE HOLD MESSAGE QUEUE. NEITHER TESTED OR SET BY EXIT.	4
			X'40'	Reserved for OTMA Callable Interface. NEITHER TESTED OR SET BY EXIT.	4
			OMHDRICN X'20'	Reserved for IMS Connect Client Type. NEITHER TESTED OR SET BY EXIT.	4
			X'10'	Reserved Client Type.	3
			X'08'	Reserved Client Type.	3

Table 47. HWSOMHDR DSECT - OTMA state data header. State data common section for Server Available and Client Bid command format (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRFL2	1	2D		CLIENT FLAGS.	3
			X'80'	OMHDRMXI - Specifies that OTMA limit the number of active messages that IMS can process concurrently. The maximum number is specified in the OMHDRTIB field.	
			X'40'	Reserved	
			X'20'	OMHDRTMO - CM1 connections time out if OTMA does not receive an ACK message in the interval specified in the OMHDRTO field.	
			X'10'	OMHDRCSC – Set by IMS Connect to indicate that synchlevel=2 (syncpoint) transactions use RRS cascaded transaction support.	
			X'08'	OMHDRSMI - This client is a member of the super member group specified in the OMHDRSMN field.	
			X'04'	OMHDR2SY - This client processes synchronous callout messages from IMS application programs.	
			X'02'	OMHDRTOQ - CM0 ACK Timeout Queue Name.	
			X'01'	OMHDRITI - Set by IMS Connect to indicate that IMS Connect supports IMS-to-IMS TCP/IP communications for OTMA messages.	
OMHDRUAV	4	2E		SAF USER ID TABLE AGING VALUE. NEITHER TESTED OR SET BY EXIT.	4
OMHDRHTS	4	32		MESSAGE RE-ASSEMBLY HASH TABLE SIZE. NEITHER TESTED OR SET BY EXIT.	4

Table 47. HWSOMHDR DSECT - OTMA state data header. State data common section for Server Available and Client Bid command format (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRSMN	4	36			
OMHDROSY	2	3A			
OMHDRODE	2	3C		Offset to the OTMA destination descriptor in the user data section of the message prefix.	
OMHDRTIB	2	3E			
OMHDRFL3	1	40		Client bid flags	
			X'80'	OMHDRRTY - MULTIRTP=Y is set for this connection. The tpipe supports multiple active resume tpipe requests.	
			X'40'	OMHDRRTN - MULTIRTP=N is set for this connection. The tpipe supports only one active resume tpipe request at a time.	
			X'10'	OMHDRSAY - SENDALTP=Y is set for this data store connection.	
			X'08'	OMHDRSAN - SENDALTP=N is set for this data store connection.	
OMHDRTO	1	41			
OMHDRTQN	8	42			
		2	ORG OMHDRORG		

Resume output for single named TPIPE for the asynchronous option of NO OPTION selection

Table 48. HWSOMHDR DSECT - OTMA state data header. State data common section for resume output for single named TPIPE for the asynchronous option of NO OPTION selection

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDCRSM_COUNT	2	2		NUMBER OF TPIPES IN THE ARRAY. Set to number of tpipes to retrieve output from a RESUME TPIPE call. Only valid value is one (1).	1

Table 48. HWSOMHDR DSECT - OTMA state data header. State data common section for resume output for single named TPIPE for the asynchronous option of NO OPTION selection (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDCRSM_TPIPEN	n	4		TPIPE ARRAY. Set to the name of the tpipe to retrieve output from a RESUME TPIPE call. Only one name is valid.	1
		2	ORG OMHDRORG		

Resume output for single named TPIPE for options of NOAUTO, SINGLE, SINGLE with WAIT, and AUTO

Table 49. HWSOMHDR DSECT - OTMA state data header. State data common section for resume output for single named TPIPE for options of NOAUTO, SINGLE, SINGLE with WAIT, and AUTO

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRRHQ	1	2	Message return options for resume tpipe calls		
			OMHDRRHQ_NOAUTO - X'00'	NOAUTO - Return all messages that are on the IMS queue when the call is received. Any messages received after the resume tpipe call is received are not sent. If no messages are on the queue, do not wait. Default option of HWSIMS00, HWSIMS01, HWSSMPL0, HWSSMPL1.	1
			OMHDRRHQ_AUTO1 - X'04'	SINGLE with WAIT - Return one message only from the queue. If no message is on the queue, wait and send the next message to arrive. This option requires that IRM_TIMER be set to X'E9' on ACK to IMS Connect from client, as an indication not to wait for any further output from IMS Connect.	
			OMHDRRHQ_AUTO - X'02'	AUTO - Return all messages that are currently on the IMS queue. After all current messages are sent, wait and send each new message as it arrives. This option requires that IRM_TIMER be set to any value other than X'E9' on ACK to IMS Connect from client, as an indication to wait for next output from IMS Connect.	1
			OMHDRRQ_ONE - X'01'	SINGLE - Return one message only from the queue. If no messages are on the queue, cancel resume tpipe request. A new RESUME TPIPE Receive sequence is required to get any subsequent messages.	1
OMHDRRHQ_MODE	1	3	Processing mode for synchronous callout messages		
			OMHDRRHQ_SYNC - X'80'	This client processes only synchronous callout messages.	
			OMHDRRHQ_SYAS - X'40'	This client processes both synchronous callout messages and asynchronous messages	
			OMHDRRHQ_CTLDATA - X'20'	This client supports callout messages containing control data.	
			OMHDRRHQ_NWSE - X'10'	This client supports output messages that contain network security information.	

Table 49. HWSOMHDR DSECT - OTMA state data header. State data common section for resume output for single named TPIPE for options of NOAUTO, SINGLE, SINGLE with WAIT, and AUTO (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDCRHQ_TPIPEN	8	4		Tpipe name. Set to the name of the tpipe to retrieve output from a RESUME TPIPE call. Only one name is valid.	1
		2	ORG OMHDRORG		

Transaction messages

Table 50. HWSOMHDR DSECT - OTMA state data header. State data common section for transaction messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRIST	1	2	OMHDRCNV X'80'	IMS STATE FLAG Conversational State.	2
			OMHDRRSP X'40'	Response Mode. Neither tested nor set by exit.	4
			OMHDRMHQ X'20'	Message from Hold Queue.	
			OMHDRROT X'08'	Message rerouted	
			OMHDRNWS X'02'	The output message for clients that issued a RESUME TPIPE call contains network security information in the security header.	
			OMHDRXP1 X'01'	For input transaction messages, X'01' indicates that a transaction expiration time is provided in the user data section at the offset specified in OMHDRSXP	
			OMHDRTOE X'01'	For resume tpipe requests and responses, X'01' indicates that a resume tpipe token is present in the OMHDRTKN field.	

Table 50. HWSOMHDR DSECT - OTMA state data header. State data common section for transaction messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRSYN	1	3	OMHDRCTD X'80'	Control data is present in the application data section of the message. This value is set by OTMA on outbound synchronous callout requests.	
			OMHDRCM0 X'40'	Commit Mode 0 Set if default for exit or the IRM requests Commit Mode 0, field IRM_F2 is set to "IRM_CMODE0."	1
			OMHDRCM1 X'20'	Commit Mode 1 Set if default for exit or the IRM requests Commit Mode 1, field IRM_F2 is set to "IRM_CMODE1."	1
			OMHDRNTX X'10'	Notify of transfer. NEITHER TESTED NOR SET BY EXIT.	4
			OMHDRSYC X'08'	This is a synchronous callout message.	
			OMHDRI2I X'04'	Set by OTMA to indicate that this messages is destined for a remote IMS system by way of an IMS Connect to IMS Connect connection.	

Table 50. HWSOMHDR DSECT - OTMA state data header. State data common section for transaction messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRSLV	1	4	OMHDRSL0 X'00'	SYNCH LEVEL Synchlevel=0 (None) Set if default for exit or the IRM requests synch level none, field IRM_F3 is not set to "IRM_CONFIRM." Synchlevel=0 is only valid for Commit Mode 1.	1
			OMHDRSL1 X'01'	Synchlevel=1 (Confirm) Set if default for exit or the IRM requests synch level confirm, field IRM_F3 is set to "IRM_CONFIRM." Synchlevel=1 is valid for Commit Modes 0 and 1.	1
			OMHDRSL2 X'02'	Synchlevel=2 (Syncpt) NEITHER TESTED NOR SET BY EXIT.	

Table 50. HWSOMHDR DSECT - OTMA state data header. State data common section for transaction messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRCFL	1	5	OMHDRSOM X'80'	Set for SENDONLY. SENDONLY is indicated in the IRM from the client.	1
			OMHDRAGN X'40'	Specifies either: • An aging value is in effect for the input message. • This is a NAK response message to a synchronous callout message that instructs OTMA to retain the undelivered message on the tpipe hold queue until it is retrieved by another resume tpipe or the message times out.	
			OMHDRRRQ X'20'	Set for REROUTE REQUEST for undeliverable CM0 output.	
			OMHDRPND X'10'	Set for PURGE NOT DELIVERABLE. SENDONLY and PURGE NOT DELIVERABLE are mutually exclusive.	
			OMHDREWC X'04'	Obsolete. EWLM is no longer supported by IMS. If this flag is specified, it is ignored by OTMA.	
OMHDRMAP	8	6		MAP NAME If client application requested that the MODname be returned, then the exit must build an RRM in front of the data being returned to the client. The MODname (OMHDRMAP) would be moved to the RRM to field RRM_MODNAME by the exit.	1

Table 50. HWSOMHDR DSECT - OTMA state data header. State data common section for transaction messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRTOK	16	E		SERVER TOKEN. NEITHER TESTED NOR SET BY EXIT.	4
		ORG OMHDRTOK + 4			
OMHDRSXP	2	12		OFFSET IN USER DATA PREFIX TO THE TRAN EXPIRATION UTC	
OMHDRSXQ	2	14		Offset in the user data section at which OTMA placed the correlation token for this synchronous callout message. This offset is specified by IMS Connect and passed to OTMA in the OMHDROSY field in the client bid.	
OMHDRUID	8	16		USERID RETURNED FOR RT REQ	
OMHDRCOR	16	1E		CORRELATOR. NEITHER TESTED NOR SET BY EXIT.	4
		ORG OMHDRCOR + 8			
OMHDRTIM	8	26		Time stamp correlator ID. On send and receive messages, this field contains the time stamp that is used to correlate responses to original input messages.	
OMHDRTKN	8	2E		Resume tpipe token. On resume tpipe requests and associated responses, this field contains the resume tpipe token.	
		ORG OMHDRTKN			
OMHDRPDR	8	36		Program name for synchronous callout messages using DL/I ICAL.	
OMHDRCID	16	2E		CONTEXT ID. NEITHER TESTED NOR SET BY EXIT.	4

Table 50. HWSOMHDR DSECT - OTMA state data header. State data common section for transaction messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMHDRLTM	8	3E		OVERRIDE LTERM NAME. Set from IRM LTERM field "IRM_LTERM."	1
OMHDRLIU	2	46		LENGTH OF IMS HEADER USER DATA. Set to the user header data length that follows. The length of this field is not included in the total length.	5
OMHDRIUD	n	48		IMS HEADER USER DATA. Variable length, set by the user.	5
	ORG OMHDRTOK+4				
OMHDRSXP	2	12	Offset at which the transaction expiration time is located in the user data section of the OTMA header	This field is set by IMS Connect	2

Resource state protocol command

The resource state protocol command is used by OTMA for two purposes: to notify the OTMA client about the current state of the IMS server resources or to send a heartbeat message to the client. IMS Connect currently ignores heartbeat messages.

The resource state protocol command is identified by X'3C' in the command type field (OMCTLTYP) of the message control information section of the OTMA header.

When IMS Connect receives a resource state protocol command, IMS Connect updates the exit interface block data store entry (HWSXIBDS with the information contained in the resource state protocol command message.

The following table summarizes the format of state data for the resource state protocol command. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 51. HWSOMHDR DSECT - OTMA state data header. State data common section for resource state protocol command format

Content	Length	Dec offset	Hex offset	Value
OMHDRLEN	2	0	X'00'	Length of the state-data section, including the length field itself.

Table 51. HWSOMHDR DSECT - OTMA state data header. State data common section for resource state protocol command format (continued)

Content	Length	Dec offset	Hex offset	Value
OMHDRSIM_STATUS	2	2	X'02'	The state of OTMA resources in the IMS system server. **X'03' - Normal state** IMS is available and processing messages normally. OTMA issues a normal state protocol command: • When an OTMA client establishes a new tpipe connection • When an OTMA client reestablishes a tpipe connection • As a heartbeat message at 60 second intervals **X'02' - Degraded state** IMS is processing messages slowly. OTMA issues a degraded state protocol command when one or more conditions indicate that IMS is not processing messages as quickly as it should. **X'01' - Unavailable state** IMS can no longer accept messages for processing. OTMA issues the unavailable state protocol command to alert the OTMA client that one or more severe conditions prevent IMS from processing OTMA messages.
OMHDRSIM_SVRFLG1	1	4	X'04'	Reserved
OMHDRSIM_SVRFLG2	1	5	X'05'	Reserved
OMHDRSIM_SVRFLG3	1	6	X'06'	Reserved
OMHDRSIM_SVRFLG4	1	7	X'07'	Flags for resource in the fourth group of unavailable resources. **X'01' - OMHDRSIM_S4FLOOD** The server is flooded with OTMA messages that are waiting to be processed and is no longer available.
OMHDRSIM_WRNFLG1	1	8	X'08'	Flags for resources in the first group of degraded resources. **X'80' - OMHDRSIM_W1FLOOD** Global flood warning for all OTMA clients **X'40' - OMHDRSIM_W1MTP** Global tpipe warning. The total number of tpipes that are being monitored by OTMA on the server is equal to or greater than the highest specified limit for any one OTMA client.
OMHDRSIM_WRNFLG2	1	9	X'09'	Reserved
OMHDRSIM_WRNFLG3	1	10	X'0A'	Reserved

Table 51. HWSOMHDR DSECT - OTMA state data header. State data common section for resource state protocol command format (continued)

Content	Length	Dec offset	Hex offset	Value
OMHDRSIM_WRNFLG4	1	11	X'0B'	Flags for resources in the fourth group of degraded resources. **X'08' - OMHDRSIM_W5MTP** The number of tpipes for this OTMA client has reached the maximum allowable number of tpipes set for this client on the MAXTP parameter of the OTMA client descriptor. No new tpipes can be created for this OTMA client until the number of tpipes drops. **X'04' - OMHDRSIM_W4MTP** The number of tpipes for this OTMA client has reached 80% of the maximum allowable number of tpipes set for this client on the MAXTP parameter of the OTMA client descriptor. **X'02' - OMHDRSIM_AWE** Message AWE reaches 80% flood. **X'01' - OMHDRSIM_W4FLOOD** Flood warning for this client only. The number of OTMA messages waiting to be processed on the server is at eighty percent of the maximum allowable number defined for the server.
OMHDRSIM_NRSFLGS	1	12	X'0C'	Other flags for non-resource related indicators. **X'80' - OMHDRSIM_HB60S** Identifies this message as a heartbeat message. The server is available and resource usage is within normal limits. Heartbeat messages are sent every 60 seconds.
Reserved	3	13	X'0D'	
OMHDRSIM_SRVNAME	16	16	X'10'	The 16 character z/OS cross-system coupling facility (XCF) member name of the OTMA server.
OMHDRSIM_CLTNAME	16	32	X'20'	The 16 character XCF member name of the OTMA client.
Reserved	20	48	X'30'	
OMHDRSIM_UTC	12	68	X'44'	UTC time for this message

Related reference

"State data section" on page 868
The state data is mandatory for any OTMA message. It immediately follows the message-control information section in the message prefix. It contains transaction-related information.

OTMA security data fields used by IMS Connect

The tables in this topic define the fields of the OTMA security data header and the order of those fields.

The numbered notes in each table are explained in "Notes to OTMA header tables" on page 281.

The DSECTs for network security information, HWSECDNDS and HWSECARDS, are generated only if you specify both of the following options in the HWSOMPFX macro:

DSECT=
Generates an individual DSECT for each section of the OTMA header. However, the HWSECDNDS and HWSECARDS DSECTS are not generated.

NETSEC_OPT=YES
Generates the HWSECDNDS and HWSECARDS DSECTs if you also specify the DSECT= option.

Common security data section for all messages

Table 52. HWS0MSEC DSECT - OTMA security data header. Security data common section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMSECLEN	2	0		SECURITY DATA LENGTH	
OMSECFLG	1	2	OMSECNON C'N'	SECURITY FLAG No RACF checking. Set to 'N' if no OTMA RACF calls are to be made.	1
			OMSECCHK C'C'	Check for transaction and command. NEITHER TESTED NOR SET BY EXIT.	4
			OMSECFUL C'F'	Check for transaction, command, and MPR Set to 'F' is OTMA is to issue RACF call.	1
OMSECFLN	1	3		LENGTH OF FOLLOWING FIELDS Set to length of USERID and GROUPID section. • Set to X'0A' if only USERID. • Set to X'14' if USERID and GROUPID. • Set to X'00' if neither USERID or GROUPID present.	1

USERID security data section for all messages

Table 53. HWSECUDS DSECT - OTMA USERID definition. Security data USERID section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMSECULN	1	0		LENGTH OF USERID FIELDS Set to length of USERID fields. The length includes this field. Set to X'09' if USERID present.	1
OMSECUTY	1	1	OMSECUXX X'02'	FIELD TYPE USERID type. Set to X'02' to identify USERID present.	1
OMSECUID	8	2		USERID Set to USERID from IRM field IRM_RACF_USERID.	1

GROUPID security data section for all messages

Table 54. HWSECGDS DSECT - OTMA GROUPID definition. Security data GROUPID section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMSECGLN	1	0		LENGTH OF GROUPID FIELDS Set to length of GROUPID fields. The length includes this field. Set to X'09' if GROUPID present.	1
OMSECGTY	1	1	OMSECGXX X'02'	FIELD TYPE GROUPID type. Set to X'03' to identify GROUPID present.	1
OMSECGRP	8	2		RACF GROUPID Set to GROUPID from IRM field IRM_RACF_GROUPID or from default GROUPID from IMS Connect configuration file.	1

UTOKEN security data section for all messages

Table 55. HWSECFDS DSECT - OTMA RACF UTOKEN definition. Security data UTOKEN section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMSECRLN	1	0		LENGTH OF UTOKEN FIELDS Set to length of UTOKEN fields. The length includes this field. Set to X'51' if user security exit issued RACF call.	1
OMSECRTY	1	1	OMSECRXX X'02'	FIELD TYPE UTOKEN type. Set to X'00' to identify UTOKEN present.	1
OMSECPRF	80	2		UTOKEN Set to UTOKEN from user security exit.	1

NETUID security data section for all messages

Table 56. HWSECDNDS DSECT - Network user ID (distinguished name) security data section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMSECDNLN	1	0		LENGTH OF NETUID FIELDS. The length of this section is variable and has a maximum value of 247. The length does not include this field.	1
OMSECDNTY	1	1	OMSECDNXX X'04'	FIELD TYPE NETUID type. Set to X'04' to indicate that a network user ID (NETUID) is present.	1
OMSECDN	1 - 246	2		The size of this field can be in the range 1 - 246 bytes. This field can contain the contents of an IRM extension that has an ID of *NETUID*.	1

NETSID security data section for all messages

Table 57. HWSECARDS DSECT - Network session ID (realm or authenticating registry) security data section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMSECARLN	1	0		LENGTH OF NETSID FIELDS. The length of this section is variable and has a maximum value of 255. The length does not include this field.	1
OMSECARTY	1	1	OMSECARXX X'05'	FIELD TYPE NETSID type. Set to X'05' to indicate that a network session ID (NETSID) is present.	1
OMSECAR	Up to 254 bytes	2		This size of this field can be in the range 1 - 254 bytes. This field can contain the contents of an IRM extension that has an ID of *NETSID*.	1

Related reference

"Security data section" on page 883
The security-data section is mandatory for every transaction or command, and is optional for OTMA protocol commands.

OTMA user data fields used by IMS Connect

The format of the user data fields in the OTMA header is defined by the HWSOMUSR DSECT in the HWSOMPFX macro and is common to all IMS Connect messages.

The numbered notes in the table are explained in "Notes to OTMA header tables" on page 281.

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSRLEN	2	0		USER DATA LENGTH. Set to length of User Data Header.	1
	2	2		ALIGN FULLWORD.	
OMUSR_DESTID	8	4		DESTINATION ID. Set to destination ID (data store) from IRM field IRM_IMSDESTID.	1
OMUSR_ORIGID	8	C		ORIGIN ID. Neither tested nor set by exit.	4

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_PORTID	8	14		PORTID. Neither tested nor set by exit.	4
OMUSR_LTOKEN	8	1C		LOGON TOKEN. Neither tested nor set by exit.	4
OMUSR_RETCODE	4	24		Communication return code. Neither tested nor set by exit.	4
OMUSR_RESCODE	8	28		Communication reason code. Neither tested nor set by exit.	4
OMUSR_RTOKEN	4	30		RESPONSE TOKEN - A(SVT) COMMUNICATION RETURN CODE. Neither tested nor set by exit.	4
OMUSR_PASSTICK	8	34		RACF PASSWORD/ PASSTICKET. Set to password from IRM field IRM_RACF_PW. This field must be cleared before passing message back to IMS Connect.	1

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_FLAG1	1	3C	OMUSR_CONV_OPT EQU X'80'	For IMS TM Resource Adapter, specifies IMS conversational transaction support for service oriented architecture composite business applications (process choreography).	4
			OMUSR_NPSOCKET X'40'	Non-persistent Socket. Neither tested nor set by exit.	4
			OMUSR_CANCID X'20'	Cancel client ID When establishing a new session, cancel client ID requests that if an existing session on the same port is using the same client ID as the new session, the existing session be terminated to allow the new session to connect.	
			OMUSR_PSOCKET X'10'	Persistent Socket. Set if persistent socket specified in IRM field IRM_SOCT has been set to IRM_SOCT_PER.	4
			OMUSR_RTALTCID X'08'	RESUME TPIPE call with an alternate client ID.	1
			OMUSR_RRDFLT X'04'	Default reroute name used.	4
			OMUSR_CANTMR X'02'	Cancel timer request. This allows you to cancel the timer (from IRM_TIMER or default configuration timer value) wait when waiting on data from the data store.	1
			OMUSR_REROUT X'01'	Reroute undeliverable output. When Commit Mode 0 (CM0) output cannot be delivered, this option causes the output to be rerouted to an alternate IMS Connect destination.	1
			OMUSR_TRAN X'00'	Transaction Socket. Set if transaction socket specified in IRM field IRM_SOCT has been set to IRM_SOCT_TRAN.	

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_FLAG2	1	3D	OMUSR_PWDTEXT X'01'	PASSTCKT field is text password. Neither tested nor set by exit.	4
			OMUSR_PWDBIN X'02'	PASSTCKT field is binary password. Neither tested nor set by exit.	4
			OMUSR_HWSPLSET X'10'	For IMS TM Resource Adapter, set if the IMS Connect protocol level is specified in the OMUSR_PROLEV field.	
			OMUSR_TRSTUSR X'80'	Trusted user can be set by exit.	
			OMUSR_F2_CIDREQ X'40'	For IMS TM Resource Adapter clients, specifies that if the ID of the current connection is a duplicate of an ID of an existing connection, IMS Connect generate a unique client ID for this connection.	
			OMUSR_F2_CIDGEN X'20'	Set when a reply message sent to IMS TM Resource Adapter by IMS Connect contains a generated client ID.	
OMUSR_FLAG3	1	3E	OMUSR_HDRCM0 X'40'	ORIGINAL SYNCHRONIZATION. Commit Mode 0. Neither tested nor set by exit.	4
			OMUSR_HDRCM1 X'20'	Commit Mode 1 (Send Commit). Neither tested nor set by exit.	4
			OMUSR_SOA X'08'	Send-only protocol with the acknowledgment option.	
			OMUSR_SOE X’80’	Send-only protocol with error option. This option can be used with OMUSR_SOO X’04’.	
			OMUSR_SOO X'04'	Send-only protocol with the serial (ordered) delivery option. This option can be used with OMUSR_SOE X’80’.	
			OMUSR_ALT_ANAK X'02'	ACK or NAK requests output to be rerouted to an alternate client ID	
			OMUSR_OM_MSG X'01'	OM message present	

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_TIMER	1	3F	OMUSR_ZERO X'E9' - X'nn' specifies no wait. OMUSR_FF specifies wait forever. See “Timeout specifications on input messages” on page 311 for a range of timer values.	OMUSR_TIMER specifies the amount of time that IMS Connect will wait for a reply from OTMA. Value is supplied by the IRM_TIMER byte, the TIMEOUT= keyword of the IMS Connect TCPIP configuration statement, or the internal IMS Connect logic.	1
OMUSR_USTAT	4	40		USTAT ADDRESS. Neither tested nor set by exit.	4
OMUSR_APPL_NM	8	44		PassTicket APPLname set to blanks or IRM value.	1
OMUSR_RRS_RCD	4	4C		z/OS Resource Recovery Services return code	

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_ARCLEV	1	50	Architectural level of the message structure of IMS TM Resource Adapter messages.		
			OMUSR_AL00 - X'00': Indicates the base architecture level IMS TM Resource Adapter		
			OMUSR_AL02 - X'02': Indicates architecture level 2, which is required to support: • Reroute of undeliverable CM0 output • RESUME TPIPE call with an alternate client ID		
			OMUSR_AL03 - X'03': Indicates architecture level 3, which is required to support IMS-to-IMS TCP/IP communications for MSC and OTMA		
OMUSR_PROLEV	1	51	When OMUSR_HWSPLSET is set, this byte specifies the protocol level of this IMS Connect instance.		
			OMUSR_PR00 - X'00'	Base protocol level	
			OMUSR_PR01 - X'01'	Currently not used	
			OMUSR_PR02 - X'02'	CM0 NOWAIT ACK support	
			OMUSR_PR03 - X'03'	Support for return codes in the request status message (RSM) from the RACROUTE user verification call.	
OMUSR_RES4	1	52		Reserved for IMS Connect usage. Neither tested nor set by exit.	3
OMUSR_RES5	4	54		Reserved for IMS Connect usage. Neither tested nor set by exit.	3

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_FLAG5	1	53	OMUSR_SOCORTIM X'80'	The message correlator value in OMHDRTIM is valid for the SendOnly transaction.	
OMUSR_RES6	4	58		Reserved for IMS Connect usage. Neither tested nor set by exit.	3
OMUSR_REROUT_NM	8	5C		Client Reroute name. Set to reroute name specified in IRM_REROUT_NM. This field occupies the same offset as the OMUSR_RT_ALTCID field.	1
	ORG OMUSR_REROUT_NM				
OMUSR_RT_ALTCID	8	5C		Alternate client ID. Tested by exit. Also requires OMUSR_ARCLEV to be set to OMUSR_AL02. This field occupies the same offset as the OMUSR_REROUT_NM field.	6
OMUSR_ADPTNM	8	64		Adapter name.	
OMUSR_DRVNM	8	6C		Input driver name.	
OMUSR_LCLIMSID	8	74		IMS ID of the IMS system sending the message.	
OMUSR_RMTICON	8	7C		The name of the remote IMS Connect connection (RMTIMSCON) as defined in the local IMS Connect.	
OMUSR_RMTIMSID	8	84		IMS ID of the IMS system receiving the message.	
OMUSR_RMTTRAN	8	8C		Transaction code that the receiving IMS system schedules to process the message.	
OMUSR_RMTUID	8	94		User ID that the receiving IMS system uses to authorize the message for transaction processing.	
OMUSR_SESTKN	8	9C		The session token used to track events during IMS Connect processing of IMS-to-IMS messages.	
	2	A4		Reserved. Align to fullword.	
OMUSR_UTC_TO	16	A6		Transaction expiration time in UTC format	
OMUSR_CORTKN	0XL40			Correlation token for synchronous callout messages	
OMUSR_CT_LEN	2	B6		Length of the correlation token	

Table 58. HWSOMUSR DSECT - user data header. User data common section for all messages (continued)

Field	Length	Hexadecimal offset	Field value	Description and settings	Note
OMUSR_CT_PSTNR	2	B8		Region ID for the synchronous callout message	
OMUSR_CT_IMSID	4	BA		IMS identifier	
OMUSR_CT_MEMTK	8	BE		OTMA TMEMBER token	
OMUSR_CT_AWETK	8	C6		OTMA message token	
OMUSR_CT_TPIPE	8	CE		OTMA TPIPE name	
OMUSR_CT_USERID	8	D6		User ID from ICAL call	
OMUSR_TRCKID_OFF	2	DE		Offset to tracking ID. Set by user message exit to point to the tracking ID copied to the user data section from one of the IRM extensions.	
OMUSR_CONTXT_OFF	2	E0		Offset to context information. This field is set by the user message exit to point to the context information copied to the user data section from one of the IRM extensions.	
	2	E2		Reserved for IMS Connect usage.	
	2	E4		Reserved for IMS Connect usage.	
OMUSR_RES6	24	E6		Reserved for IMS Connect usage.	
Reserved	2	FE		Align to fullword.	

Related reference

"User data section" on page 886
The user-data section of the OTMA message prefix is variable length and follows the security-data section. It can contain any data.

Notes to OTMA header tables

The following notes correspond to the numbers shown in the tables that describe the fields of the OTMA header.

- Note **1**: Set by READ routine of user-written exit.
- Note **2**: Set by IMS Connect.
- Note **3**: Reserved fields.
- Note **4**: Set by IMS Connect and not analyzed by user exit.
- Note **5**: User-defined area.
- Note **6**: Analyzed by user exit.
- Note **7**: Only an output flag.

Chapter 18. IMS Connect protocols

IMS Connect provides several different transaction protocols.

The protocols include:

- Conversational support
- Send only
- RESUME TPIPE/Receive for asynchronous output
- Socket connections
- Asynchronous output support

Transaction restrictions and limitations

The restrictions and limitations for transactions can differ depending on the specific transaction type.

The following is a list of restrictions and limitations of specific transactions:

- IMS Fast Path, conversational, and nonrecoverable transactions must be issued using commit mode 1. This is a restriction of IMS OTMA.
- Non-response transactions can be sent to IMS Connect using the SENDONLY option and must be issued using commit mode 0 on a transaction or persistent socket.

Commit mode and synchronization level definitions

IMS Connect supports the following commit modes and synchronization levels.

Commit mode 0

Commit mode 0 (CM0) is also called commit-then-send. CM0 is supported on both persistent and transaction sockets and supports only synch level CONFIRM.

If your version of IMS Connect supports protocol level X'02', you can set the IRM_F1_NOWAIT flag on inbound CM0 messages. This flag indicates that the client will not wait for the final timeout message from IMS Connect after it acknowledges receipt of the message.

Commit mode 1

Commit mode 1 (CM1) is also called send-then-commit. CM1 is supported on both persistent and transaction sockets and supports synch levels NONE, CONFIRM, and SYNCH.

For CM1 input messages, always specify a timeout value for IMS to wait for the ACK and NAK responses from IMS Connect. Until an ACK or NAK response is received by IMS, the IMS dependent regions remains occupied and continues to hold the necessary database locks. Valid timeout intervals are from 0 to 255 seconds and are specified on the ACKTO parameter of the DATASTORE configuration statement.

You can view the current timeout values by issuing the IMS Connect commands VIEWHWS or VIEWDS, or you can issue the MVS command QUERY MEMBER.

Synch Level=NONE

The synchronization level specifies the level of acknowledgment for each transaction. If a transaction is specified with Synch Level=NONE, no acknowledgment is required from the client. The database changes are still committed if the output message is sent to IMS Connect, but not to the client. However, if OTMA is unable to deliver the output message to IMS Connect, the input and output message are discarded, the database changes are backed out, and the IMS application terminates and returns with a 119 ABEND.

© Copyright IBM Corp. 1974, 2020

Synch Level=CONFIRM
If a transaction is specified with Synch Level=CONFIRM, the client is required to send an acknowledgment to signal to IMS Connect whether the output message was successfully (ACK) or unsuccessfully (NAK) processed by the client.

The processing of CONFIRM is dependent on the type of commit mode that you specify:

- If Synch Level=CONFIRM is requested with CM0, and the client responds with ACK, IMS dequeues the output message. If the client returns a NAK response, the output message is requeued in IMS for later retrieval by the client.
- If Synch Level=CONFIRM is requested with commit mode 1, and the client responds with ACK, the database changes are committed. If the client responds with NAK, the database changes are backed out and the output message is discarded by IMS.

Synch Level=SYNCH
If a transaction is specified with Synch Level=SYNCH, two phase commit processing is required. Use Synch Level=SYNCH when multiple participants are involved in sync point processing. Synch Level=SYNCH is managed through z/OS Resource Recovery Services.

Related concepts

"Socket connections" on page 300
IMS Connect provides three kinds of client TCP/IP connection protocols, which are called *sockets*. The TCP/IP sockets define how IMS Connect manages client TCP/IP connections when IMS Connect sends a disconnect message.

"IMS Connect protocol level" on page 284
The IMS Connect protocol level identifies which transaction features and modes IMS Connect is configured to support.

IMS Connect protocol level

The IMS Connect protocol level identifies which transaction features and modes IMS Connect is configured to support.

Any successfully processed output message from the IMS Connect message exit sent to a client contains a segment called the complete status message (CSM). The third byte of the CSM is a status flag. A flag value of X'10' indicates that the fourth byte of the CSM is a second flag byte which contains the current IMS Connect protocol level.

X'00'
All features and modes are enabled except CM0 ACK NOWAIT transaction support for send-and-receive transactions.

X'01'
This protocol level is reserved.

X'02'
CM0 ACK NOWAIT transaction support is enabled in addition to all other support.

X'03'
Return codes from the RACROUTE user verification call are supported in the request status message (RSM).

If the CSM indicates that CM0 ACK NOWAIT support is enabled, the client can then submit a CM0 ACK NOWAIT transaction by setting the IRM_F1_NOWAIT flag in the IMS request message (IRM) segment of the input message sent to IMS Connect and specifying an IRM_TIMER value of X'E9'. The NOWAIT flag indicates the client will not wait for a final timeout message from IMS Connect after acknowledging the successful receipt of the transaction.

Related reference

"Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines" on page 218

Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

IMS Connect conversational support

A conversational program is a message processing program (MPP) that processes transactions made up of several steps. The MPP does not process the entire transaction at once.

The conversational support for IMS Connect includes having conversational transactions that let you retain uninterrupted connection (continuity) for messages coming from a given client. Typically, a conversation is terminated when the message is sent and dequeued and the application program has placed blanks in the SPA, or the conversation is terminated when a COMMIT CONFIRMED messaged is received from the client. For conversational support for IMS Connect, conversations require a send-then-commit mode and are nonrecoverable.

Requirement: IMS support for SOA composite business applications requires that all iterations of a given IMS conversation be processed by the same IMS Connect and same IMS. More specifically, all iterations of a given IMS conversation must use the same conversational ID, port number, IMS Connect, and data store. This can be accomplished by using the same connection factory for every iteration of a conversation. Using the same connection factory for each iteration ensures that the same host name, port number, and data store will be used. The IMS TM Resource Adapter client must also save and reuse the same IMSInteractionSpec convID property to ensure that the same Conversational ID is used for each iteration of that conversation.

In addition, different physical socket connections, represented by IMSTCPIPManagedConnection instances, can be used for each iteration of an IMS conversation, as long as these socket connections meet the above criteria. Having originated from the same connection factory, these connections would be members of the same managed connection pool.

These requirements preclude the use of Sysplex Distributor or the SHAREPORT keyword on the PORT statement of the z/OS TCP/IP profile when you use the IMS support for SOA composite business applications.

OTMA conversational protocols

The following examples show the OTMA conversational protocols.

Send-then-commit, sync level=none

The send-then-commit flow sends IMS output before IMS completes synchronization-point (hereafter referred to as sync-point) processing.

To use the send-then-commit flow, specify commit Mode 1 in the state-data section of the message prefix.

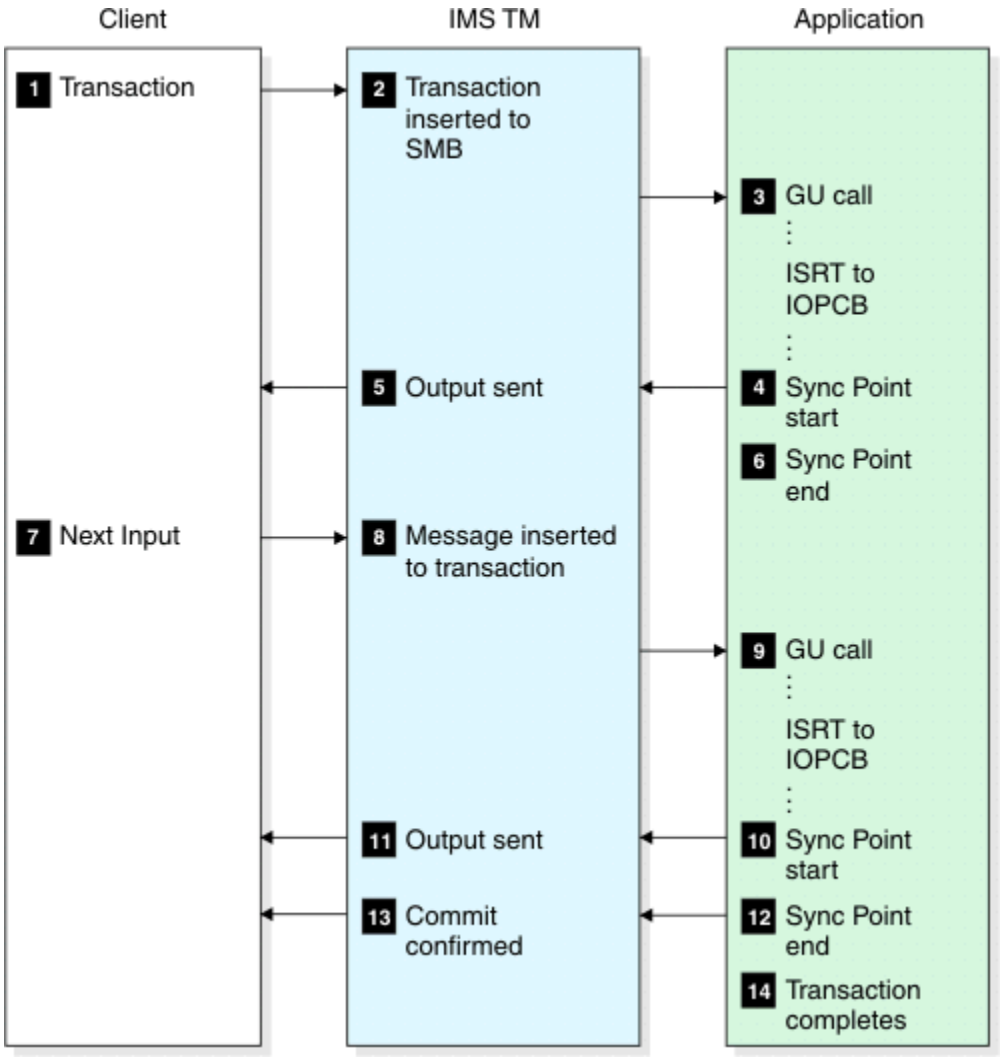


Figure 26. Send-then-commit, sync level=none flow for OTMA conversational protocols

The sample flow shown in the above figure assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as NONE in the state-data section of the message prefix. Therefore, IMS does not request a response (an ACK) when sending output.

Send-then-commit, sync level=confirm

The send-then-commit flow assumes no synchronization for the transactions as they are processed by IMS.

The following figure shows a flow in which all transactions are confirmed as they are received (each message requests a response).

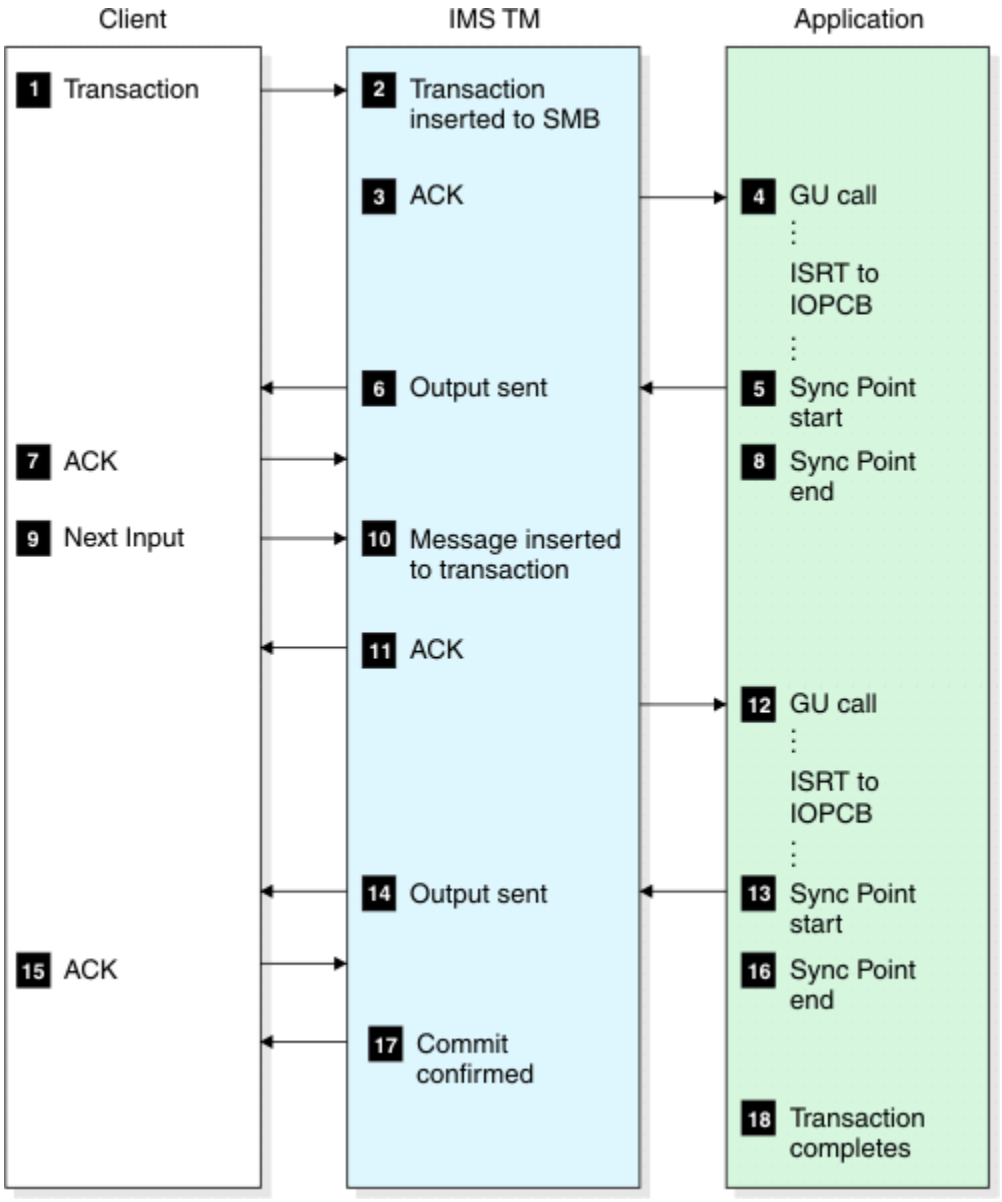


Figure 27. Send-then-commit, sync level=confirm flow for OTMA conversational protocols

The sample flow shown in the above figure assumes the following:

- Commit mode 1 is specified in the state-data section of the message prefix.
- The transaction pipe is not synchronized.
- The synchronization level is specified as Confirm in the state-data section.

IMS Connect conversational protocols

IMS Connect conversation protocols use the send-then-commit protocol. The following examples show variations of the conversational protocols with sync levels of none and confirm.

Send-then-commit, sync level=none, transaction terminated from the program

The send-then-commit flow sends IMS output before IMS completes sync-point processing.

To use the send-then-commit flow, specify commit mode 1 in the state-data section of the message prefix.

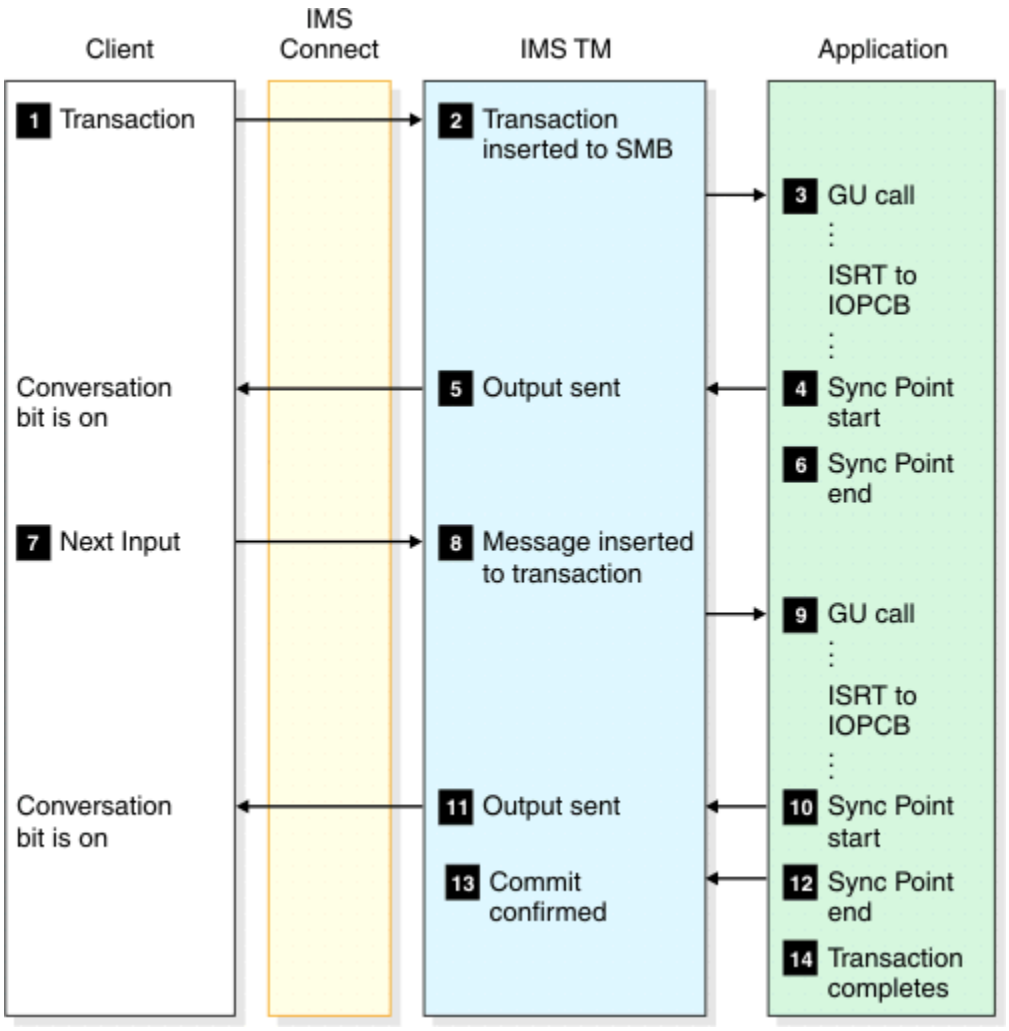


Figure 28. Send-then-commit, sync level=none (transaction terminated from program) flow

The sample flow shown in the above figure assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as NONE in the state-data section of the message prefix. Therefore, IMS does not request a response (an ACK) when sending output.
- The transaction is terminated from the program.
- IMS Connect will close the socket as soon as Commit confirmed has been sent by IMS.

Send-then-commit, sync level=none, transaction terminated from the client

The send-then-commit flow sends IMS output before IMS completes sync-point processing.

To use the send-then-commit flow, specify commit mode 1 in the state-data section of the message prefix.

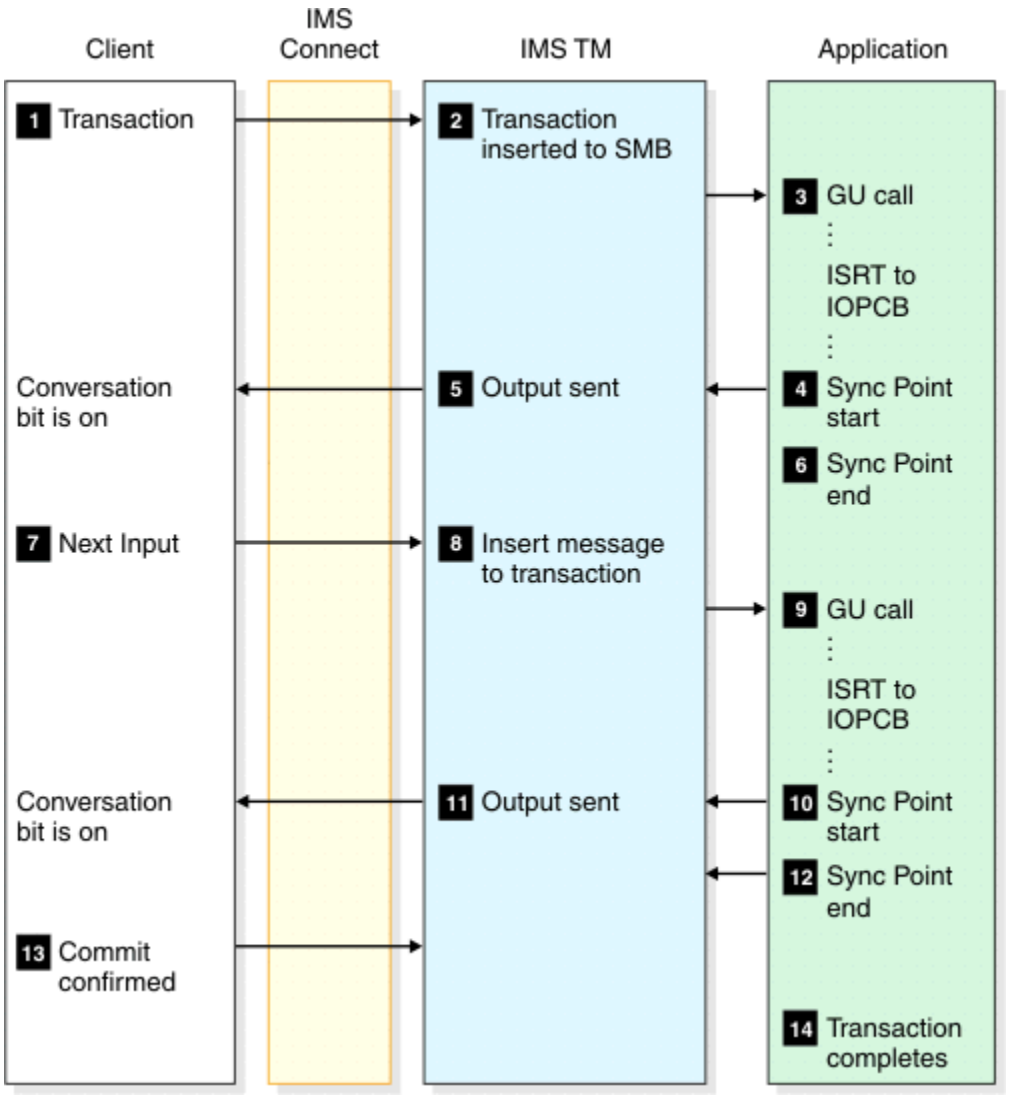


Figure 29. Send-then-commit, sync level=none (transaction terminated from client) flow

The sample flow shown in the above figure assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as NONE in the state-data section of the message prefix. Therefore, IMS does not request a response (an ACK) when sending output.
- The transaction is terminated from client.

Send-then-commit, sync level=confirm, ACK response

This send-then-commit flow assumes no synchronization for the transactions as they are processed by IMS.

The following figure shows a flow in which all transactions are confirmed as they are received (each message requests a response).

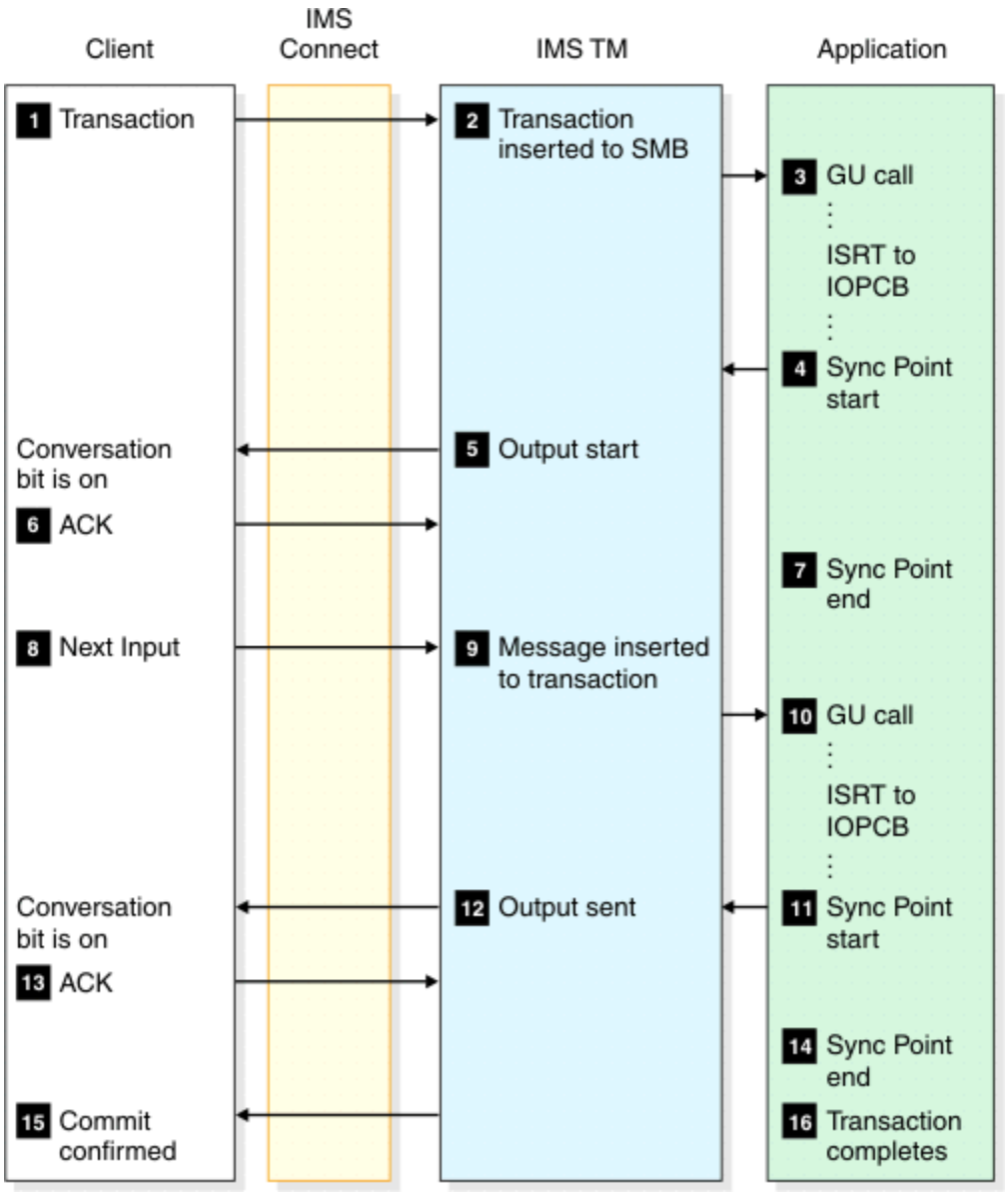


Figure 30. Send-then-commit, sync level=confirm (ACK response) flow

The sample flow shown in the preceding figure assumes the following:

- Commit mode 1 is specified in the state-data section of the message prefix.
- The transaction pipe is not synchronized.
- The synchronization level is specified as Confirm in the state-data section.
- ACK can be replied to by a remote workstation before the check response requested bit.

Send-then-commit, sync level=confirm, NAK response

This send-then-commit flow assumes no synchronization for the transactions as they are processed by IMS.

The following figure shows a flow in which all transactions are confirmed as they are received (each message requests a response).

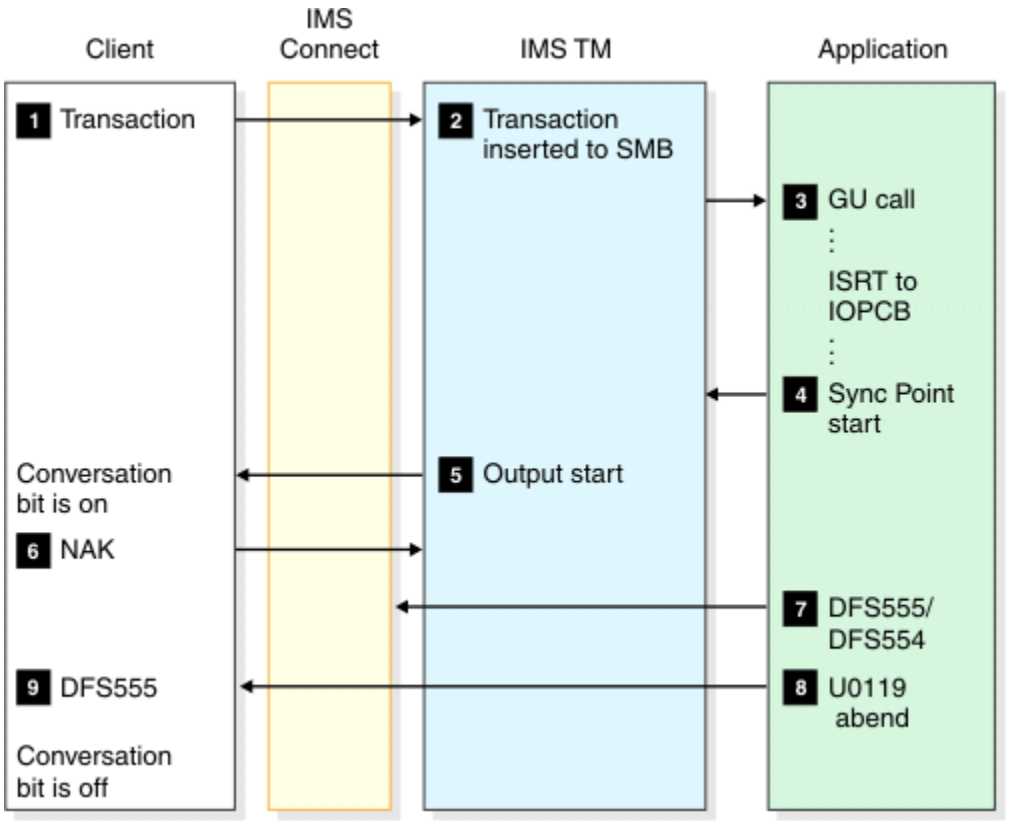


Figure 31. Send-then-commit, sync level=confirm (NAK response) flow

The sample flow shown in the preceding figure assumes the following:

- Commit mode 1 is specified in the state-data section of the message prefix.
- The transaction pipe is not synchronized.
- The synchronization level is specified as Confirm in the state-data section.
- NAK can be replied to by either IMS Connect or a remote workstation before the check requested bit.
- If the client forgets to send the NAK/ACK before it closes the socket, IMS Connect will send the NAK to IMS and it will cause a U0119 abend.

Purging undeliverable commit-then-send output

You can configure OTMA to purge commit-then-send (commit mode 0) IOPCB output when the output cannot be returned to the OTMA client application that initiated the transaction.

When configured, if the OTMA client requests the purge function, OTMA dequeues and discards the undeliverable commit mode 0 (CM0) IOPCB output from the IMS output queue. The purge function is requested on a message-by-message basis.

When the purge function is not specified, IMS stores undeliverable commit-then-send IOPCB output on the asynchronous hold queue of the OTMA tpipe associated with the client application that submitted the original input message. The output message remains on the hold queue for later retrieval by using a RESUME TPIPE call.

You can specify the purge function on either CM0 or commit mode 1 (CM1) input messages. However, when specified on CM1 input messages, IMS purges only CM0 IOPCB output, such as might be generated by a program-to-program switch. For example, if you specify the purge function on a CM1 transaction input that does a program switch to a second transaction and the first transaction does an insert to the IOPCB, the purge function applies only to the subsequent transactions that insert to the IOPCB.

Both user-written applications and IMS TM Resource Adapter applications on either persistent sockets or transaction sockets can request the purge function.

Specifying the purge function for undeliverable commit-then-send output

If you are using either the HWSSMPL0 or the HWSSMPL1 IMS Connect user message exit, you can enable the purge function for undeliverable commit-then-send output by specifying the IRM_F3_PURGE flag (X'04') in the IRM_F3 field on certain types of input messages.

You can specify the purge function on the following input messages from a client application:

- A SEND of a commit-then-send (CM0) transaction.
- A SEND of a send-then-commit (CM1) transaction. A purge request on CM1 input applies only to any CM0 output that the CM1 input generates.
- A SEND of a NAK response to CM0 output.

Restriction: You cannot specify the purge function and the reroute function at the same time. If both functions are specified, the output messages are neither purged nor rerouted from the original output queue and OTMA issues message DFS2407W.

When IMS purges undeliverable commit-then-send output

If the purge function is specified, IMS purges commit-then-send (commit mode 0) output in certain circumstances only.

When the purge function is specified, IMS purges commit-then-send (commit mode 0) output when:

- IMS Connect receives a NAK response from the client application
- IMS Connect cannot deliver the output to the client application
- OTMA cannot deliver the output to IMS Connect

When IMS Connect receives a NAK response from the client or cannot deliver the output to the client, IMS Connect notifies OTMA to discard the output message from the IOPCB queue. When OTMA cannot deliver the output to IMS Connect, OTMA discards the output without waiting for notification from IMS Connect.

If an IMS Connect STOPCLNT command is issued for a client ID that specifies the purge function, the reply message is purged.

IMS does not support the purge function for the following types of output:

- IMS application output to ALTPCBs. Even if the purge function is specified for the IMS application CM0 output, the purge function does not apply to inserts to ALTPCBs.
- CM1 output. Any output from a CM1 transaction that is undeliverable is already discarded and the transaction is backed out.
- Output associated with a send-only transaction. Output for send-only transactions is routed directly to the asynchronous hold queue.
- Output associated with a RESUME TPIPE call input message. A RESUME TPIPE call, by definition, requires and guarantees that the output is delivered.

The purge function, multiple-message output, and NAKs

After receiving a NAK response from a client application for one of multiple related messages for which the purge function is specified, IMS Connect issues purge requests for the remaining output messages without attempting to deliver them to the client application and OTMA discards them from the IOPCB.

How IMS Connect issues the purge requests differs slightly depending on whether the multiple messages are generated by a single application program or by program-to-program switches.

If there are multiple commit then send output messages for the same application program on the IOPCB and the client issues a NAK response for one of the messages, IMS Connect sends the purge request to OTMA. IMS Connect also sends purge requests to OTMA for the remaining output messages on the IOPCB without attempting to deliver the output to the client application.

If there are multiple commit then send output messages generated by program-to-program switches on the IOPCB and the client issues a NAK response for one of the messages, IMS Connect passes a purge

request to OTMA and then generates additional purge requests for any other related output messages currently on the IOPCB queue. If program-to-program switches generate related output messages after the initial NAK response was received, IMS Connect issues purge requests for them as well without passing the output to the client application.

Rerouting commit-then-send output

You can configure IMS to reroute commit-then-send (commit mode 0) IOPCB output to an alternate OTMA tpipe hold queue for retrieval.

Normally, if IMS cannot return commit mode 0 (CM0) output to the application client, the output is routed to the tpipe hold queue associated with the client application that submitted the original message; however, if you request the reroute function, IMS reroutes the output to either a user-specified tpipe hold queue or the default tpipe hold queue HWS$DEF. Whether the reroute tpipe is a user-specified tpipe or the default tpipe, the reroute tpipe is always associated with the tmember of the original tpipe.

The reroute function can be used for managing output generated by send-only transactions and for managing output that cannot be delivered to the original client because the connection timed out or failed.

The reroute function is also useful when IMS TM Resource Adapter (formerly known as IMS Connector for Java) is used with shareable persistent sockets. The IMS TM Resource Adapter automatically generates the client ID when connecting to IMS Connect. Consequently, the client ID is unknown to the client applications, which need the client ID to retrieve the CM0 output.

You can specify the reroute function in either CM0 or CM1 input messages. However, in the case of CM1, IMS can only reroute the CM0 output, such as might be generated by a program to program switch.

Both user-written applications and IMS TM Resource Adapter applications on either persistent sockets or transaction sockets can request the reroute function.

Restrictions: The reroute function is not supported for:

- CM1 output messages
- Output resulting from a RESUME TPIPE call
- Output resulting from an insert to an ALTPCB

Specifying the reroute function for commit-then-send output

You enable the reroute function for commit-then-send output by setting a flag in the IRM header of your input message or by coding your user-written user message exit to set the appropriate flag in the OTMA state data.

If you are using either the HWSSMPL0 or the HWSSMPL1 IMS Connect user message exit, you can enable the reroute function for commit-then-send output by specifying the IRM_F3_REROUT flag (X'08') in the IRM_F3 field for the following input messages from a client application:

- A SEND of a commit-then-send (CM0) transaction
- A SEND of a CM0 send-only transaction from a user-written client application
- A SEND of a send-then-commit (CM1) transaction (A reroute request on CM1 input applies only to any CM0 output that the CM1 input generates)
- A SEND of a NAK response to CM0 output

Restriction: You cannot specify the purge function and the reroute function at the same time. If both functions are specified, the output messages are neither purged nor rerouted from the original output queue and OTMA issues message DFS2407W.

Specifying a destination for rerouted output

You can define the reroute destination by specifying a reroute request name. Output can only be rerouted to tpipes associated with the same tmember. Specifying a destination for rerouted output is optional.

If a client application requests that output be rerouted, but does not identify a reroute destination by specifying a tpipe name, the default reroute destination is tpipe HWS$DEF.

You can define the reroute destination by specifying a reroute request name in one or more of the following places:

- The RRNAME= keyword in the IMS Connect DATASTORE configuration file
- An IMS Connect user message exit
- The IRM_REROUT_NM in the fixed IRM format of an input message associated with a SEND/RECEIVE request from a client application
- The IRM_REROUT_NM in the fixed IRM format of a NAK message from a client application

You can specify a different reroute destination in a NAK response message than is specified by an initial input message; however, doing so can cause problems for transactions that generate multiple output messages. If a different reroute destination is specified in a NAK response message and multiple output messages are generated by the initial input message, OTMA reroutes only the message that triggered the NAK response to the destination specified in the NAK response message. After receiving the NAK response, OTMA automatically reroutes any subsequent output messages for the same transaction to the destination that was specified on the initial input message.

For more information about:

- The RRNAME= keyword of the IMS Connect DATASTORE configuration file, see *IMS Version 15 System Definition*.
- Coding IMS Connect user message exits, see *IMS Version 15 Exit Routines*.

Related reference

"IMS Connect message structures" on page 213
TCP/IP clients using the z/OS program call interface communicate with IMS Connect by using an IMS request message (IRM) header on each input message. The IRM header is used on input messages from IMS Connect client application programs to communicate protocol options to IMS Connect. The IRM header is mapped by the IMS Connect HWSIMSCB macro.

DATASTORE statement (System Definition)

When IMS reroutes commit-then-send output

If the reroute function is specified, IMS reroutes commit-then-send (CM0) output only in certain circumstances.

When the reroute function is specified, IMS reroutes commit-then-send (CM0) output if:

- IMS Connect cannot deliver the output to the client application.
- IMS Connect receives a NAK response from the client application.
- OTMA cannot deliver the output to IMS Connect.
- IMS inserts output for send-only transactions to the IOPCB.

When IMS Connect cannot deliver the output or when IMS Connect receives a NAK response, IMS Connect notifies OTMA to reroute the output message to the alternate destination.

In the case of a NAK response from the client application, OTMA reroutes the output if the NAK is in response to an IMS application insert to the IOPCB. If the NAK is in response to the output related to a RESUME TPIPE call, IMS does not reroute the output.

In the event of a communication failure between OTMA and IMS Connect, OTMA reroutes the commit-then-send output only if the original input message requested the reroute function.

For send-only transactions, when the reroute function is specified OTMA always reroutes the output.

If IMS Connect receives a disconnect notification on the TCP/IP READ for an ACK or NAK response to an output message, IMS Connect requests that OTMA reroute the CM0 output only if the input message specified the reroute function. If the client disconnects or times out prior to IMS Connect receiving the output message from OTMA, the CM0 output message is rerouted only if the input message specified the reroute function.

The reroute function, multiple-message output, and NAK responses

If a transaction produces multiple output messages, OTMA reroutes the first output message that triggers a NAK response and any subsequent output messages for the same transaction that are on the IOPCB at the time the NAK response message is received.

OTMA reroutes subsequent output messages for the same transaction that arrive to the IOPCB after the initial reroute only after triggering another NAK response.

If a transaction initiates a program-to-program switch and IMS Connect receives a NAK response to the first output message from an application program, OTMA reroutes output messages sent by a secondary application program after a program-to-program switch only if they are already on the output queue when the initial NAK response message is received or if the original input message specifies a reroute destination. If the original input message does not specify a reroute destination, OTMA does not reroute undeliverable output generated after a program-to-program switch.

If a send-then-commit (CM1) transaction message does a program-to-program switch to a second CM0 transaction message and the first transaction does an insert to the IOPCB, IMS reroutes only the second or subsequent CM0 messages that insert to the IOPCB. IMS reroutes the CM0 output of a CM1 input message only if the client application requests reroute in the original CM1 transaction message.

Recoverable IMS transactions

This topic contains scenarios for running recoverable transactions in the IMS Connect environment.

For each of the following scenarios:

- OTMA will have deleted the input message.
- Requeuing of the input message will not occur.
- For CM1 (send-then-commit), none of the output is placed (ENQUEUED) in the IMS queue.

Only commit mode 0 (CM0, also referred to as commit-then-send) is treated as recoverable; CM1 is not recoverable. With the use of CM0, IMS Connect creates a separate TPIPE for each client that uses CM0. This TPIPE remains in IMS, so a fixed client name is highly recommended for each client that intends to use CM0.

The combination of commit mode and Sync level is critical. The following scenarios describe the different uses and the results.

- With CM1 and SYNC LEVEL = NONE:

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends the message to the client, and any ACK/NAK from the client in response to the output message would become an error because the ACK/NAK are not expected and IMS Connect would have received a message from the client with no application data.
- With CM1 and SYNC LEVEL = CONFIRM:

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it to the client, and an ACK from the client will result in the successful completion of the application. This scenario works as expected.

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it back to the client, and a NAK from the client will result in an IMS MPP 119 abend and an IMS message, DFS555. The 119 abend will back out the database changes, and both the input and

output messages are discarded. The result would be as if the system had never seen the transaction, and a reentry of the transaction would be necessary.

- With CM0 and SYNC LEVEL = CONFIRM:

  The input message is processed by IMS, the application program commits the changes, and an output message is sent back to IMS Connect. IMS Connect sends it to the client, and an ACK response from the client results in IMS dequeuing the output, and represents the successful completion of transaction processing. CM0 forces the Synch level to Confirm. This scenario works as expected.

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it back to the client, and a NAK from the client will result in the database changes not being backed out. The input message is discarded and the output message is requeued to the IMS queue for representation. These output messages will be moved to the hold asynchronous queue by OTMA, and will be retrievable only with the RESUME TPIPE, RECEIVE and ACK process.

Recommendation: To run recoverable transactions in the IMS Connect environment, use CM0 and SYNC LEVEL = CONFIRM, and use a single unique CLIENT_ID for each client that uses CM0 and SYNC LEVEL = CONFIRM

Send-only protocol

Client application programs use the send-only protocol to submit commit-then-send (CM0) input messages to IMS in rapid succession without requiring the client application to wait for a response. The send-only protocol is designed for fast, high volume input. Any error that occurs in IMS Connect or IMS is not returned to the client.

Clients configured to support synchronous callout requests also use the send-only protocol to return responses from an external data or service provider to an IMS application program that issued the synchronous callout request and that is waiting for the response in an IMS dependent region.

The output generated by IMS in response to send only input is stored on an asynchronous hold queue associated with tpipe used by the client application and can be retrieved later by issuing a RESUME TPIPE call.

The send-only protocol does not return errors that occur in IMS Connect or IMS. If clients want to receive responses for errors that occur in IMS Connect, use the send-only with error protocol.

If send-only input messages must be processed by IMS in the order in which they are sent by the client, the send-only protocol offers two options that can help ensure that IMS receives the messages in the order in which IMS Connect receives them from the client application:

- Send-only protocol with the acknowledgment option
- Send-only protocol with the serial delivery option

When the send-only protocol is used to return responses from a data or service provider external to the IMS installation, the send-only message does not contain a transaction code, but rather the data requested by the IMS application or, in the event there was a problem processing the callout request, error codes.

The following figure shows an example of the CM0 send-only protocol flow when the commit confirmed flag is on. The CM0 flow, also known as the IMS standard flow, enqueues IMS output before sending it to the client. However, in this case for non-response transactions, the client does not expect any output from IMS.

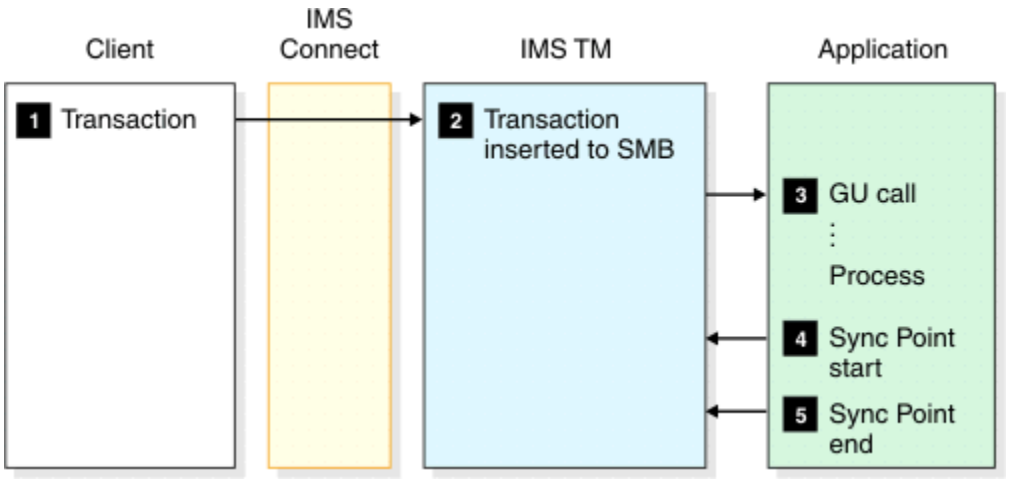


Figure 32. Send-only protocol flow

The preceding sample flow assumes the following:

- CM0 is specified in the state-data section of the message prefix.
- The transaction bit and the commit confirmed bit is specified in the control-data section of the message prefix.

Send-only with acknowledgment protocol

Use the send-only protocol with acknowledgment (SNDONLYA) to ensure that send only transaction inputs are enqueued by IMS in the same order that they are submitted by the client application program.

SNDONLYA messages execute a send-only interaction for a non-response mode, non-conversational transaction. If the host application terminates without issuing an ISRT to the IO PCB, no DFS2082 messages are returned to the client. SNDONLYA also specifies that IMS Connect must indicate in its reply that an ACK or NAK of the output is required from the client. The SNDONLYA interaction must use CM0.

When SNDONLYA is specified, the client application receives an ACK response message from OTMA for each input message successfully enqueued by IMS. All other output generated by the send only transaction is sent to the asynchronous hold queue.

Before sending the next send only input message, the client application must wait for and process the ACK response. Because subsequent input messages sent by the client application to the same tpipe are not sent until the preceding input message has been enqueued, transaction messages are enqueued in IMS in the order in which they were sent.

You can select the send-only protocol with the acknowledgment option by specifying L in the IRM_F4 field of the HWSSMPL0 and HWSSMPL1 user message exit IRM format.

The send-only protocol with the acknowledgment option and the send-only protocol with the serial delivery option are mutually exclusive. If both are specified, the send only with acknowledgment protocol takes effect.

The send-only protocol with the acknowledgment option is not supported by HWSJAVA0.

Related concepts

Send-only with error protocol
Use the send-only with error (SNDONLYE) protocol to receive responses for errors that occur in IMS Connect.

Send-only protocol with serial delivery protocol
When the send-only protocol with the serial delivery option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.

Send-only protocol for synchronous callout responses

IMS Connect clients return responses to synchronous callout requests from IMS application programs by using the send-only protocol.

Send-only with error protocol

Use the send-only with error (SNDONLYE) protocol to receive responses for errors that occur in IMS Connect.

When the send-only protocol is used, IMS Connect (ICON) does not return responses to the client for any error that occurs in IMS or IMS Connect. The client does not wait for any response from IMS Connect.

If you use the send-only with error protocol, IMS Connect returns an error response to the client and closes the socket connection if it detects an error during IMS Connect inbound transaction message processing. If IMS Connect detects no error, no response is returned to the client.

IMS Connect does not send any response for errors in the IMS system back to the client with the send-only with error protocol.

If you specify the send-only with error protocol to an ICON that does not support this protocol, the ICON ignores the unsupported send-only with error option and treats the request as a send/receive protocol.

It is recommended to use a separate thread to receive error responses so the main thread could continue to send transactions to IMS Connect and maintain high volume input.

You can enable the send-only with error protocol by specifying the IRM_F4_SNDONLYE C'J' bit in the IRM_F4 field for messages that use HWSSMPL0 and HWSSMPL1 user message exit IRM format.

The send-only with error protocol option and the send-only protocol with serial delivery option can be used together.

Related concepts

Send-only with acknowledgment protocol
Use the send-only protocol with acknowledgment (SNDONLYA) to ensure that send only transaction inputs are enqueued by IMS in the same order that they are submitted by the client application program.

Send-only protocol with serial delivery protocol
When the send-only protocol with the serial delivery option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.

Send-only protocol for synchronous callout responses
IMS Connect clients return responses to synchronous callout requests from IMS application programs by using the send-only protocol.

"Send-only protocol with serial delivery protocol" on page 298
When the send-only protocol with the serial delivery option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.

Related reference

"Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines" on page 218
Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

Send-only protocol with serial delivery protocol

When the send-only protocol with the serial delivery option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.

You can select the send-only protocol with the serial delivery option by specifying X'10' for IRM_F3_ORDER in the IRM_F3 field of the HWSSMPL0 and HWSSMPL1 user message exit IRM format.

The send-only protocol with the serial delivery option and the send-only protocol with the acknowledgment option are mutually exclusive. If both are specified, the send only with acknowledgment protocol takes effect.

The send-only protocol with the serial delivery option is not supported by HWSJAVA0.

The following requirements apply to the send-only protocol when the serial delivery option is used.

- All transactions that use the send-only protocol with the serial delivery option must be defined as SCHDTYP=SERIAL. If the transactions are defined as SCHDTYP=PARALLEL, OTMA cannot guarantee that IMS will process them serially.
- Send-only transactions that must be delivered serially relative to one another must use the same tmember and tpipe connection, including transactions of the same transaction type that must be delivered serially with respect to all other transactions of the same type. If a transaction is sent on multiple tmember and tpipe connections, OTMA cannot guarantee that IMS will process them serially.

The the send-only protocol with serial delivery option and the send-only protocol with error option can be used together.

Related concepts

Send-only with acknowledgment protocol
Use the send-only protocol with acknowledgment (SNDONLYA) to ensure that send only transaction inputs are enqueued by IMS in the same order that they are submitted by the client application program.

Send-only with error protocol
Use the send-only with error (SNDONLYE) protocol to receive responses for errors that occur in IMS Connect.

Send-only protocol for synchronous callout responses
IMS Connect clients return responses to synchronous callout requests from IMS application programs by using the send-only protocol.

Send-only protocol for synchronous callout responses

IMS Connect clients return responses to synchronous callout requests from IMS application programs by using the send-only protocol.

The IMS Connect client specifies the send-only protocol for synchronous callout responses in the IRM_F4 field of the IRM prefix of the response message.

The send-only protocol for synchronous callout responses can include an acknowledgement to the callout response, or the acknowledgement can be disabled so the client does not need to switch to receive state after sending the response to IMS Connect.

If the client requires IMS to return an acknowledgement after IMS receives the callout response, the client can specify an L in the IRM_F4 field of the IRM message prefix. When L is specified, if delivery of the response is successful, IMS Connect returns a CSM (ACK). If delivery of the response is unsuccessful, IMS Connect returns an RSM (NAK). The client must issue an additional receive to retrieve the CSM or RSM, which could affect the performance of the client, but does not impact the return of the response to the waiting IMS application.

If the client does not require IMS to return an acknowledgement, the client can specify an M in the IRM_F4 field to disable the acknowledgements. When the acknowledgements are disabled, IMS Connect does not return an RSM message to the client.

The send-only message contains no transaction code and can contain either the response data or error information for the IMS application program. Send-only messages used for callout responses do not generate any type of output.

For user-written IMS Connect clients, code the callout response by specifying the following field values in the IRM of the response message:

- IRM_ARCH = X'03' (IRM_ARCH3)
- IRM_F0 = X'10' (IRN_F0_NAKRSN), if a NAK reason code is sent with the error response

- IRM_F0= X'20' (IRM_F0_SYNCNAK), if no NAK reason code is sent with the error response
- IRM_NAK_RSNCDE = 2 byte hexadecimal extended error code
- IRM_F4 = L (IRM_F4_SYNRESPA) or M (IRM_F4_SYNRESP)
- IRM_CORTKN = 40 byte correlation token (CORTKN) from original callout request

Related concepts

Send-only with acknowledgment protocol
Use the send-only protocol with acknowledgment (SNDONLYA) to ensure that send only transaction inputs are enqueued by IMS in the same order that they are submitted by the client application program.

Send-only with error protocol
Use the send-only with error (SNDONLYE) protocol to receive responses for errors that occur in IMS Connect.

Send-only protocol with serial delivery protocol
When the send-only protocol with the serial delivery option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.

Socket connections

IMS Connect provides three kinds of client TCP/IP connection protocols, which are called *sockets*. The TCP/IP sockets define how IMS Connect manages client TCP/IP connections when IMS Connect sends a disconnect message.

The three socket types provided by IMS Connect are:

- Persistent
- Transaction
- Non-persistent

Persistent sockets

A *persistent* socket is a connection between the client and IMS Connect that remains connected until either the client or IMS Connect specifically make a disconnect request. A persistent socket can exist across multiple transactions.

There are two ways that the client can force a termination:

- By sending IMS Connect a disconnect request.
- By changing the socket type to "transaction" for the last transaction entered, such as a logoff transaction.

IMS Connect can also terminate the connection when an error occurs.

The IMS Connect user message exits HWSSMPL0, HWSSMPL1, HWSSOAP1, and HWSJAVA0 support the use of persistent sockets.

A persistent socket supports both commit mode 1 (CM1 or send then commit) and commit mode 0 (CM0 or commit then send) processing.

In the following cases, IMS Connect generates a unique client ID for a connection from IMS TM Resource Adapter on a persistent socket:

- The IMS TM Resource Adapter requests that IMS Connect check for duplicate client IDs and the client ID provided on a new connection request is a duplicate of the client ID of an existing connection.
- The IMS TM Resource Adapter passes a blank client ID to IMS Connect.

Related concepts

“Socket processing for transactions” on page 303

For a transaction on either a transaction socket or persistent socket, the client application must always issue a TCP/IP READ following all TCP/IP SENDs.

Related tasks

"Setting socket types for IMS TM clients" on page 301
For IMS Connect clients that access IMS TM, client code controls the socket settings, and the IMS Connect user message exits and the user initialization exit enforce the socket settings.

Related reference

User message exit routines HWSSMPL0 and HWSSMPL1 (Exit Routines)

IMS TM Resource Adapter user message exit routine (HWSJAVA0) (Exit Routines)

SOAP Gateway exit routine (HWSSOAP1) (Exit Routines)

Transaction sockets

A *transaction* socket is a connection between the client and IMS Connect that remains connected for a single transaction or IMS conversation. The connection can be terminated only by IMS Connect, either when IMS itself terminates, or when an error occurs.

A transaction socket supports both commit-mode-1 (CM1 or send then commit) and commit-mode-0 (CM0 or commit then send) processing.

Related concepts

"Socket processing for transactions" on page 303
For a transaction on either a transaction socket or persistent socket, the client application must always issue a TCP/IP READ following all TCP/IP SENDs.

Related tasks

"Setting socket types for IMS TM clients" on page 301
For IMS Connect clients that access IMS TM, client code controls the socket settings, and the IMS Connect user message exits and the user initialization exit enforce the socket settings.

Non-persistent sockets

A *non-persistent* socket maintains a connection for a single input-and-output pair to IMS Connect.

IMS Connect terminates the connection after sending the output to the client for non-conversational and conversational transactions. If three exchanges of input and output occur, the disconnect is issued three times, one for each output from IMS Connect.

Restriction: The HWSSMPL0, HWSSMPL1, HWSSOAP1, and HWSJAVA0 user message exit routines do not support non-persistent sockets, nor does IMS TM Resource Adapter.

Related tasks

"Setting socket types for IMS TM clients" on page 301
For IMS Connect clients that access IMS TM, client code controls the socket settings, and the IMS Connect user message exits and the user initialization exit enforce the socket settings.

Setting socket types for IMS TM clients

For IMS Connect clients that access IMS TM, client code controls the socket settings, and the IMS Connect user message exits and the user initialization exit enforce the socket settings.

The client selects the socket connection type by setting a flag in the IMS request message (IRM) header, in the field IRM_SOCT. The IRM_SOCT flag values are shown in the following table.

Table 59. IRM_SOCT flags

Flag	Definition	Socket type
IRM_SOCT_PER	X'10'	Persistent
IRM_TRAN	X'00'	Transaction

Table 59. IRM_SOCT flags (continued)

Flag	Definition	Socket type
IRM_SOCT_NONPER	X'40'	Non-persistent

The IRM_SOCT flag must be set for each message that is sent to IMS Connect from an IMS Connect client that accesses IMS TM.

Recommendation: Set all messages that are associated with a single transaction to the same socket type. If you do not, unexpected results can occur, as described in the following examples:

- If the first message of a conversational transaction is set to persistent, and the last message is set to transaction, then the socket connection will be terminated following the last message.
- If one of the messages in the middle of the conversational transaction set the socket type to transaction, and the IMS transaction terminates for some reason, then IMS Connect will disconnect the socket. This is because "transaction" was the last known socket type.

The user message exits determine the socket type, then move the socket type information to the user data section of the OTMA message header that they return to IMS Connect. To transfer the socket type information to the OTMA message header, the user exits set the OMUSR_FLAG1 field, which is mapped by the HWSOMUSR DSECT of the HWSOMPFX macro, with one of the following flags as shown in the following table:

Table 60. OMUSR_FLAG1 flags

Flag	Definition	Socket type
OMUSR_PSOCKET	X'10'	Persistent
OMUSR_TRAN	X'00'	Transaction
OMUSR_NPSOCKET	X'40'	Non-persistent

Related reference

“IRM structures for IMS Connect client messages” on page 213
IMS Connect expects all client messages that it receives to start with a four byte total length field, followed by an IMS request message (IRM) header, followed by the message data segments.

“OTMA user data fields used by IMS Connect” on page 274
The format of the user data fields in the OTMA header is defined by the HWSOMUSR DSECT in the HWSOMPFX macro and is common to all IMS Connect messages.

Socket connections for IMS-to-IMS TCP/IP communications

Socket connections for IMS-to-IMS TCP/IP communications, which connect two IMS Connect instances, can be persistent or non-persistent.

You specify the persistence of the socket connection by using the PERSISTENT keyword of the RMTIMSCON configuration statement.

If the connection is used for MSC, the connection must be persistent. If you specify PERSISTENT=NO, IMS Connect issues a warning message and sets the value to PERSISTENT=YES.

If the connection is used for OTMA, the connection can be either persistent or non-persistent; however, to avoid the risk of a proliferation of tpipes on the remote IMS system under high-volume circumstances, specify PERSISTENT=YES.

Socket termination scenarios

When a WTOR **CLOSEHWS** or similar z/OS MODIFY or IMS type-2 command is issued, IMS Connect closes all send socket connections that are in the CONN state and quiesces all send socket connections that are in the RECV state. For socket connections that are in the RECV state, IMS Connect waits indefinitely for

the ACK or NAK acknowledgement from the remote IMS Connect. IMS Connect terminates only after all connections that are in the RECV state receive an acknowledgement.

When a WTOR **CLOSEHWS** FORCE or similar z/OS MODIFY or IMS type-2 command is issued, IMS Connect closes all send socket connections. If any send socket connections are in the RECV state, IMS Connect sends a NAK to the local OTMA and OTMA reroutes the sent messages that did not receive an acknowledgement to the timeout queue.

If IMS Connect terminates abnormally, OTMA reroutes any sent messages that did not receive an acknowledgement to the timeout queue.

For MSC socket connections, if communication with MSC stops because IMS or SCI terminated normally or abnormally, IMS Connect terminates all MSC logical links on the affected MSC physical links. IMS Connect closes all send and receive sockets used by the terminated MSC logical links. The status in IMS Connect of the affected MSC physical link is set to DISCONNECT.

Related concepts

"Reserving send sockets for IMS-to-IMS communication" on page 306
You can reserve IMS Connect send sockets for use by IMS-to-IMS TCP/IP connections by specifying the RESVSOC parameter on the RMTIMSCON configuration statement.

IMS Connect definition and tailoring (System Definition)

Related reference

CLOSEHWS command (Commands)

UPDATE IMSCON TYPE(CONFIG) command (Commands)

RMTIMSCON statement (System Definition)

Socket processing for transactions

For a transaction on either a transaction socket or persistent socket, the client application must always issue a TCP/IP READ following all TCP/IP SENDs.

The exceptions are for a TCP/IP SEND of SENDONLY or a TCP/IP SEND of an ACK with IRM_TIMER set to NO_WAIT (X'E9' char Z), which is issued in response to a READ of a RESUME_TPIPE single request.

The following scenarios describe transactions on a transaction socket. For transactions on a persistent socket, the process is the same as transactions on a transaction socket. However, the client application and IMS Connect do not disconnect. Also, the client application will receive a return code of X'28' if there is a timeout. The return code states a disconnect is not required.

For a Commit mode 0, Synch Level Confirm, non-conversational transaction on a transaction socket, the following scenario occurs:

1. The client application issues a SEND to send the transaction data to IMS Connect.
2. IMS Connect returns the output to the client application.
3. The client application receives the output, sends an ACK, and must issue a READ to receive the next output or the timeout notification.
4. IMS Connect issues a timeout notification with the return code of either X'20' or X'24' for a transaction socket, or an X'28' for a persistent socket. IMS Connect will disconnect the socket for the X'20' and X'24' return codes, and will keep the connection for the X'28' return code.
5. The client application issues a disconnect for return codes X'20' and X'24'. The client can issue a disconnect for return code X'28' or send in the next input.

For a CM1, Synch Level Confirm, non-conversational transaction on a transaction socket, the following scenario occurs:

1. The client application issues a SEND of the transaction data to IMS Connect.
2. IMS Connect returns the output to the client application.
3. The client application receives the output, sends an ACK or NAK, and issues a READ.
4. After an ACK is sent, the client receives one of the following responses:

- Deallocate commit if the IMS transaction completes successfully.
- A DFS message if the IMS transaction failed.
- A timeout notification with a return code of X'20' or X'24' for a transaction socket or a return code of X'28' for a persistent socket. The client application is required to issue a disconnect for return codes X'20' and X'24'.

5. The client application issues a disconnect.

For a CM1, Synch Level Confirm, conversational transaction on a transaction socket, the following scenario occurs:

1. The client application issues a SEND to send the transaction data to IMS Connect.
2. IMS Connect returns the output to the client application.
3. The client application receives the output, sends an ACK, and issues the next input. The client continues SEND, READ, ACK until the transaction is complete.
4. IMS Connect issues an RSM deallocate commit, deallocate abort, or a timeout notification. The timeout notification returns either X'20' or X'24', which indicates that IMS Connect will disconnect.
5. The client application issues a disconnect.

Managing the number of sockets

IMS Connect supports from 50 to 65,535 sockets at one time. Within that range, you can define both a maximum number of allowable sockets and the point at which IMS Connect issues warnings when the number of sockets approaches the maximum.

The total number of sockets that IMS Connect supports across all TCP/IP ports at the same time is set by the MAXSOC parameter in the TCPIP configuration statement of the IMS Connect configuration member. If the number of sockets reaches the maximum, IMS Connect refuses any new connections and issues message HWSS0771W. After the number of connections falls below the MAXSOC value, IMS Connect resumes accepting connections.

Because IMS Connect uses one socket on each TCP/IP port for listening, the maximum number of physical connections that IMS Connect supports is the MAXSOC value less the number of TCP/IP ports. For example, if you specify MAXSOC=80 and have five TCP/IP ports, 75 physical connections can be made.

By default, the MAXSOC parameter sets the maximum number of sockets at 50.

As the number of sockets approaches the maximum allowable number, IMS Connect issues warnings at a warning threshold set by the WARNSOC parameter and, if the number of sockets continues to rise, at increments set by the WARNINC parameter thereafter. Both the WARNSOC parameter and the WARNINC parameter are specified on the TCPIP configuration statement in the IMS Connect HWSCFGxx PROCLIB member.

You can display the current values of the warning threshold and the warning increment, as well as the current number of open sockets for an instance of IMS Connect, by issuing any of the following commands:

- IMS Connect WTOR command **VIEWHWS**
- IMS Connect z/OS command **QUERY MEMBER**
- IMS Connect type-2 command **QUERY IMSCON TYPE(CONFIG)**

You can also display the current number of sockets on a given port by issuing any of the following commands:

- IMS Connect WTOR command **VIEWPORT**
- IMS Connect z/OS command **QUERY PORT**
- IMS Connect type-2 command **QUERY IMSCON TYPE(PORT)**

Even when there are no active client connections on a port, when displaying the current number of sockets, IMS Connect ports always show one open socket: the port listen socket.

The IMS Connect SSL port shows at least two sockets: the port listen socket and a socket that represents the SSL-related file descriptor. When the IMS Connect SSL socket receives its first connection, an additional SSL-related file descriptor is created, so that the IMS Connect SSL socket shows four sockets for the first client connection: the port listen socket, two file descriptors, and the client connection socket.

Related concepts

IMS Connect definition and tailoring (System Definition)

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

Limits on the number of sockets set

The maximum number of sockets supported by an IMS Connect instance is affected by four parameters: MAXSOC, MAXSOCKETS, MAXFILEPROC, and FILEPROCMAX. IMS Connect can open ports only if the MAXSOC parameter is compatible with the UNIX System Services MAXFILEPROC parameter, or if a valid override is in effect.

The **MAXSOC=** parameter on the HWSCFGxx member of the IMS PROCLIB data set specifies the total number of sockets that IMS Connect will support. However, the actual number of client sockets available is less than the total MAXSOC number because IMS Connect uses one socket per port as a listener socket, the optional product IMS Connect Extensions for z/OS uses two sockets, and other vendor products might also use sockets.

The z/OS UNIX System Services parameters **MAXSOCKETS** and **MAXFILEPROC** are set on the BPXPRMxx member of the SYS1.PROCLIB data set and affect the total number of sockets as follows:

- The **MAXSOCKETS** parameter sets a limit for the total number of sockets in a system. To learn more, see MAXSOCKETS.
- The **MAXFILEPROC** parameter sets the maximum number of sockets for any user ID (this is the IMS Connect address space user ID). To learn more, see MAXFILEPROC.

FILEPROCMAX is a security setting parameter that can restrict the number of sockets for a user ID just as MAXFILEPROC can. It is originally specified on the ADDUSER statement for RACF. It can be altered by the ALTUSER (Alter user profile) command.

IMS Connect cannot open any ports unless its **MAXSOC** parameter value ***is less than or equal to*** the **MAXFILEPROC** parameter value, or to the **FILEPROCMAX** value if the FILEPROCMAX field of the OMVS segment of a user's profile is used to override MAXFILEPROC. The following methods briefly describe how to ensure MAXSOC compatibility. For details, see TCPIP statement (System Definition).

1. Before you start IMS Connect, check that the **MAXSOC** value is less than or equal to the **MAXFILEPROC** value. Decrease the MAXSOC value if necessary. Alternatively, to increase the MAXFILEPROC value, consult with your UNIX System Services administrator.
2. If IMS Connect is running, check the MAXFILEPROC parameter value for IMS Connect with the following UNIX commands:
   - To determine the PID (process ID) for IMS Connect, issue: `D OMVS,V`.
   - To check the MAXFILEPROC value for IMS Connect, issue the following command and include the PID from the previous step: `D OMVS,L,PID=`.

   If the MAXFILEPROC value is lower than the MAXSOC value, use the ALTUSER command to assign an equal or greater value to the **FILEPROCMAX** field of the OMVS segment of the user ID that is used to start IMS Connect. To learn more, see Defining IMS Connect security (System Definition).
3. This method is not recommended but is retained for historical context: Grant UNIX System Services superuser privileges to IMS Connect by using the RACF ALTERUSER command to assign an OMVS segment with a UID of 0 to the user ID of the IMS Connect started task. To learn more, see Superusers in z/OS UNIX.

Messages related to socket limits

The following IMS Connect and UNIX System Services messages can help you determine socket limit status.

If incompatible MAXSOC and MAXFILEPROC or FILEPROCMAX values are detected, or if IMS Connect is not granted superuser privileges, IMS Connect does not open any ports and issues the following message:

- `HWSP1415E TCP/IP SOCKET FUNCTION CALL FAILED; F=SETRLIMI, R=-1, E=139, M=SDOT`

If IMS Connect successfully opens ports, the following messages are issued if the number of sockets on an open port reaches or approaches a specified limit.

- When the **MAXSOC** parameter limit is reached, IMS Connect issues message HWSS0771W as follows:
  - `HWSS0771W LISTENING ON PORT=portid FAILED; R=rc, S=sc, M=mc`
- When the number of sockets approaches the **MAXFILEPROC** parameter value, UNIX System Services issues message BPXI040I. For example, if the MAXFILEPROC limit is 400, the following message is issued:
  - `BPXI040I PROCESS LIMIT MAXFILEPROC HAS REACHED 85% OF ITS CURRENT 400.`

    Note that message BPXI040I is displayed only if LIMMSG is set to SYSTEM or ALL in the SYS1.PARMLIB(BPXPRMxx) data set or by using the SETOMVS command.
- When the **MAXFILEPROC** parameter value is reached, IMS Connect issues messages HWSP1415E and HWSS0771W as follows:
  - `HWSP1415E TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4, R=-1, E=124, M=SDCO`
  - `HWSS0771W LISTENING ON PORT=portid FAILED; R=rc, S=sc, M=mc`

To learn more about socket limit messages, see Socket number warnings (Communications and Connections).

Related information

ALTUSER (Alter user profile)

BPXPRMxx (z/OS UNIX System Services parameters)

Reserving send sockets for IMS-to-IMS communication

You can reserve IMS Connect send sockets for use by IMS-to-IMS TCP/IP connections by specifying the RESVSOC parameter on the RMTIMSCON configuration statement.

The number of sockets specified on RESVSOC parameters count against the total number of sockets that an instance of IMS Connect can have open at a time, as specified on the MAXSOC parameter of the TCPIP configuration statement. Consequently, the total value of all RESVSOC parameters on all configuration statements cannot exceed the value specified on the MAXSOC parameter.

The RESVSOC parameter reserves only send sockets. Receive sockets cannot be reserved.

If your connection is an IMS-to-IMS connection that is used for MSC, each MSC logical link that uses the TCP/IP connection requires two sockets: one for sending data and one for receiving data. The RESVSOC parameter reserves only the send sockets. Therefore, connections to another instance of IMS Connect that are used for MSC use twice as many sockets than are specified on the RESVSOC parameter.

If your connection is an IMS-to-IMS TCP/IP connection that is used for OTMA, each connection uses only a single send socket. Therefore, the number of sockets reserved by the RESVSOC parameter represents the number of sockets that is used for each connection.

Related concepts

IMS Connect definition and tailoring (System Definition)

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

Related reference

RMTIMSCON statement (System Definition)

RMTCICS statement (System Definition)

Socket number warnings

IMS Connect issues warning message HWSS0772W when the number of sockets reaches the default warning threshold of 80 % of the maximum allowable number of sockets set by the MAXSOC parameter.

Socket number warnings

If the number of sockets continues to increase past 80 %, IMS Connect issues a new HWSS0772W message at a default increment of every 5 % increase in the number of sockets. When the number of sockets reaches the maximum allowable number, IMS Connect refuses any new connections and issues message HWSS0771W.

You can set the warning threshold by using the WARNSOC parameter on the TCPIP statement of the IMS Connect HWSCFGxx PROCLIB member. You can set the warning increment by using the WARNINC parameter on the TCPIP statement.

To prevent insignificant fluctuations in the number of sockets from flooding the console with messages, until the warning mechanism is reset, IMS Connect issues a HWSS0772W messages only once for the warning threshold and only once for each warning increment thereafter.

IMS determines the reset threshold by subtracting either twice the value of the WARNINC parameter or 5 % from the value of the MAXSOC parameter, whichever results in a lower reset threshold. When the reset threshold is reached, IMS Connect resets the warning mechanism and issues message HWSS0773I.

By default, IMS Connect resets the warning mechanism when the number of sockets falls to 70 % of the value of the MAXSOC parameter, which is the default value of the MAXSOC parameter less 10 %, which is two times the default value of the WARNINC parameter.

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

Related information

HWSS0771W (Messages and Codes)

HWSS0772W (Messages and Codes)

HWSS0773I (Messages and Codes)

Resolving duplicate client IDs

If a failure occurs between IMS Connect and the IMS Connect client, either because the connection failed or the client terminated unexpectedly, and IMS Connect does not detect the failure before the client attempts to reconnect with the same client ID, IMS Connect prevents the client from reconnecting because the client ID is a duplicate of the ID associated with the original failed connection.

To resolve duplicate client ID conditions after a failure you can have the incoming connection cancel the original client ID or you can cancel the IMS Connect message timer for the socket connection that the client was originally connected to. Canceling the client ID is the easier of these two options.

To decrease the chances of a duplicate client ID condition occurring in the first place, you can specify a small KeepAlive interval for all the socket connections on a given port. Specifying a small KeepAlive value helps IMS Connect detect and clean up failed client connections earlier, potentially before the client attempts to reconnect; however, a small KeepAlive value can also increase network traffic.

IMS TM Resource Adapter clients can also have IMS Connect generate unique client IDs for incoming connections, thereby avoiding the possibility that an incoming connection from IMS TM Resource Adapter might specify an ID that is a duplicate of an ID that is being used by an existing connection.

Canceling connections that have duplicate client IDs

When an incoming connection to IMS Connect specifies a client ID that is a duplicate of a client ID used by another existing connection, you can cancel the existing connection so that IMS Connect accepts the new connection. When a client ID is canceled, IMS Connect discards the existing connection, regardless of its state, and issues message HWSS0743I DUPLICATE CLIENT ID TERMINATED.

Canceling a connection with a duplicate client ID is supported for clients that use either persistent sockets or transactions sockets with either commit-then-send (CM0) or send-then-commit (CM1) transactions.

If a duplicate client ID is canceled, the previous session, if one exists, is cleaned up and the new request is completed. Then, if the socket type is a persistent socket, the session is maintained and if the socket is a transaction socket, the session is terminated.

For user-written IMS Connect client applications, you can cancel duplicate client ID connections in the following ways:

- By coding the IMS Connect client application to specify X'80' (IRM_F3_CANCID) in the IRM_F3 field of the user section of the IRM message header mapped by the HWSIMSCB macro when the client establishes a new connection with IMS Connect.
- By coding any of the following exit routines to specify X'20' (OMUSR_CANCID) in the OMUSR_FLAG1 field of the user data section of the OTMA message header mapped by the HWSOMPFX macro.

If IMS Connect has already detected the error and terminated the original connection, IMS Connect ignores any specification to cancel a duplicate client ID.

Canceling the message timer to resolve a duplicate client ID

You can resolve a duplicate client ID condition by canceling the IRM timer on the original failed client connection; however, the steps required to cancel a timer, which include issuing ACKs, SENDs, and READS, and connecting and disconnecting, make canceling the duplicate client ID the recommended method for resolving duplicate client ID conditions.

Auto-generated IDs for IMS TM Resource Adapter

For users of the IMS TM Resource Adapter, IMS Connect can generate a unique user ID if the user ID provided by IMS TM Resource Adapter on a new connection request is a duplicate of a user ID already in use by an existing connection.

After IMS Connect generates a user ID, the generated ID is used in all replies returned to IMS TM Resource Adapter.

When an IMS TM Resource Adapter client requests that IMS Connect generate a user ID, the OMUSR_FLAG2 field is set to X'40' (OMUSR_F2_CIDREQ) in the OTMA message prefix. When IMS Connect returns a reply that includes a generated user ID, OMUSR_FLAG2 is set to X'20' (OMUSR_F2_CIDGEN).

Related tasks

“Canceling a message timer” on page 319
User-written IMS Connect client applications can cancel the active message timer when waiting on output from the data store.

Related reference

“Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines” on page 218
Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

“OTMA user data fields used by IMS Connect” on page 274

The format of the user data fields in the OTMA header is defined by the HWSOMUSR DSECT in the HWSOMPFX macro and is common to all IMS Connect messages.

IMS Connect override for the z/OS TCP/IP KeepAlive interval

IMS Connect can override the default KeepAlive interval that is defined in z/OS for TCP/IP socket connections.

The KeepAlive function, which is a function provided by the TCP/IP protocol, can detect certain socket error conditions by sending a KeepAlive packet on sockets that have been inactive for a specified interval. For example, the KeepAlive function can detect sockets that are no longer valid because the client has abruptly disconnected without informing IMS Connect.

By default, IMS Connect accepts the specification set in the z/OS layer for TCP/IP sockets; however, if a KeepAlive interval set by z/OS is large, as might be the case for installations trying to reduce network traffic, the large interval can delay the detection of an invalid socket by IMS Connect.

To help detect error conditions earlier on IMS Connect sockets, you can specify a smaller KeepAlive interval for IMS Connect ports. Each port defined to IMS Connect can specify a different KeepAlive interval. The KeepAlive interval specified for each port applies to all sockets that use that port.

Typically, a network manager determines if IMS Connect sockets require a different KeepAlive interval from the value defined by z/OS for the TCP/IP stack. After the KeepAlive intervals for IMS Connect are determined, an IMS System Programmer updates either the PORT or DRDAPORT parameters in the IMS Connect HWSCFGxx PROCLIB member with the KeepAlive intervals.

For IMS Connect clients that access IMS TM, KeepAlive intervals can be specified by using the KEEPAV parameter of the PORT keyword in the TCPIP configuration statement in the HWSCFG PROCLIB member.

For IMS Connect clients that access IMS DB, KeepAlive intervals can be specified by using the KEEPAV parameter of the DRDAPORT keyword in the ODACCESS configuration statement in the HWSCFG PROCLIB member.

The range of valid values for the KEEPAV parameter is defined by the TCP/IP protocol and is from 1 to 2 147 460 seconds. A KEEPAV value of zero accepts the KeepAlive interval set in the z/OS layer for the TCP/IP stack. KEEPAV=0 is the default.

You can display the current KeepAlive interval for by issuing any of the following commands:

- **VIEWHWS**
- **VIEWPORT**
- **QUERY MEMBER**
- **QUERY PORT**

An IMS Connect-defined KeepAlive interval is not supported by ports defined by using the PORTID parameter, by “LOCAL” ports, or by SSL ports. If the KEEPAV parameter is specified for any of these ports, IMS Connect abends during initialization.

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

TCP/IP failures

If active IMS Connect ports unexpectedly lose their connection to the TCP/IP services provided by the TCP/IP network or the z/OS TCP/IP stack, IMS Connect automatically reconnects the ports when the TCP/IP services becomes available again.

To detect when the TCP/IP services become available, IMS Connect continues listening on all active ports. When communication with the TCP/IP services is resumed, IMS Connect reconnects the ports and issues message HWSS0780I.

If communication between a port and the TCP/IP services was terminated by a command, IMS Connect does not automatically resume communication with the TCP/IP services if they should become available.

IMS Connect timeout specifications

You can specify timeout intervals to IMS connect for various stages of the communication process and for various types of interactions.

How you specify each timeout interval differs depending on whether you are using an IMS Connect client that accesses IMS DB, an IMS Connect client that access IMS TM, or an IMS application program that sends output to a remote IMS system by way of OTMA and IMS-to-IMS TCP/IP communications.

Related concepts

"IMS Connect client support" on page 142
As a TCP/IP server and a message router for IMS, IMS Connect provides access to IMS TM, IMS DB, and the CSL Operations Manager (OM). The client support provided by IMS Connect differs, depending on which type of access the IMS Connect client needs.

"IMS Connect support for IMS-to-IMS TCP/IP communications" on page 150
IMS Connect manages the TCP/IP connections and protocols for IMS systems that communicate with each other across a TCP/IP network.

Timeout specifications for IMS DB clients

For IMS Connect clients that access IMS DB, you can specify timeout values on two parameters in the ODACCESS configuration statement.

The ODBMTMOT parameter controls the amount of time that IMS Connect waits for both:

- A response message on connections with ODBM
- An initial input message after a socket connection is established on connections with a client application

The PORTTMOT keyword controls how long IMS Connect keeps an existing connection open after the client stops sending input.

Related reference

ODACCESS statement (System Definition)

Timeout specifications for IMS TM clients

For IMS Connect clients that access IMS TM, you can specify timeout values in the TCP/IP and DATASTORE configuration statements, as well as in the IMS Request Message (IRM) header of input messages.

In the TCPIP configuration statement, you can set the following limits on the amount of time IMS Connect waits in the following stages of communication:

- How long IMS Connect keeps a connection open if the client does not send any input after the connection is first established. The TIMEOUT parameter sets this limit.
- How long IMS Connect keeps a connection that is in RECV state open after the prior client interaction completes. This limit can be set by the IDLETO parameter in the TCPIP configuration statement or by CREATE IMSCON commands.
- How long IMS Connect waits for a response from IMS before IMS Connect notifies the client of the timeout and returning the socket connection to a RECV state. This limit is also set by the TIMEOUT parameter and is overridden by input messages that specify a timeout interval in the IRM.

The idle connection timeout value (IDLETO) applies only to ports that are used for communication with IMS TM. IMS TM ports are defined by the PORT or PORTID parameter in the TCPIP statement or as PORTTYPE(REG) by the CREATE IMSCON command. The IDLETO value can be set in the TCPIP configuration statement or by using the IMS type-2 CREATE IMSCON TYPE(PORT), UPDATE IMSCON TYPE(PORT), or UPDATE IMSCON TYPE(CONFIG) command.

In the DATASTORE configuration statement, the ACKTO keyword controls how long OTMA waits for acknowledgements from IMS Connect before rerouting the output to a tpipe hold queue.

The timeout values specified on the TCPIP and DATASTORE configuration statements can also be specified in the IRM message header on messages received from IMS TM clients.

The IRM can also be used to specify how long the client waits for output after issuing a resume tpipe request.

Related reference

DATASTORE statement (System Definition)

TCPIP statement (System Definition)

CREATE IMSCON TYPE(PORT) command (Commands)

UPDATE IMSCON commands (Commands)

Timeout specifications on input messages

Each and every input message from the IMS Connect client can set a different timeout value in the IRM_TIMER field of the fixed portion of the IMS request message (IRM) header.

Set the IRM_TIMER value to an appropriate wait time for IMS to return data to IMS Connect.

The settings for the IRM_TIMER is enforced as described in the following list:

1. If the IRM_TIMER is set at X'00', the following default values are used:
   - The default for all RESUME_TPIPE is two seconds.
   - The default for all RESUME_TPIPE non-single ACK is .25 seconds.
   - The value of the TIMEOUT parameter in the IMS Connect TCPIP configuration statement for all others.
2. X'FF' and X'01' - X'9E' are used only when requested.
3. X'E9' (char Z) NO_WAIT means do not wait for any IMS output. NO_WAIT is not valid on some Client SENDs. Because IMS Connect does not wait for output from IMS, on a transaction socket connection, IMS Connect disconnects the socket; and on a persistent socket connection, IMS Connect requests the next input from the client rather than disconnect the socket. If NO_WAIT is used, it is enforced as follows:
   - There is a two second delay for:
     - RESUME_TPIPE request
     - conversational transaction code
     - conversational data
     - ACK or NAK associated with a conversational transaction
     - non-conversational transaction code
   - A .25 second delay for each of the following is used:
     - an ACK or NAK associated with a non-conversational transaction commit mode one confirm
     - an ACK or NAK associated with a RESUME_TPIPE with Asynch output options AUTO or NOAUTO
     - an ACK or NAK associated with non-conversational transaction commit mode zero confirm
   - NO_WAIT can be used for the following:
     - a SENDONLY
     - an ACK or NAK associated with RESUME_TPIPE with Asynch output option SINGLE
     - an ACK or NAK associated with non-conversational transaction commit mode zero confirm with the IRM NOWAIT flag

Misuse of X'E9' can result in one of the following problems:

1. The socket disconnects.
2. An output message to the client on a transaction socket is lost.

3. A hang condition occurs between the client and IMS Connect or IMS Connect and OTMA. For example, the client can be in a READ state waiting for output from IMS Connect while IMS Connect is in a READ state waiting for input from the client and OTMA is in READ state waiting for acknowledgment.
4. The deallocate commit or deallocate abort notification for CM1 SynchLevel=Confirm is lost.
5. Other unpredictable conditions occur.

To determine the appropriate wait time for IMS to return data to IMS Connect, consider the following guidelines:

- For a client SEND of transaction code and data or data only, the IRM_TIMER value should be set to reflect the amount of time IMS Connect should wait for the output from IMS.

  Recommendation: Do not use a timer value of X'E9' except for messages with the NOWAIT flag set on the IRM.
- If the client application knows that the last message received is the last output message to the client for the transaction, set the IRM_TIMER to X'01' (.01 of a second) for a client SEND of ACK or NAK. The IRM_TIMER of X'01' is the smallest value that can be set for non-RESUME TPIPE ACK messages. However, if the ACK message is associated with an output from a RESUME TPIPE call, do **not** set the IRM_TIMER value to X'E9' (character Z).
- For a client SEND of a RESUME TPIPE call, the timer value can be set as follows:

  AUTO option
  X'FF' for dedicated output device, or any X'00' to X'9E' values for non-dedicated output device

  NOAUTO option
  any value other than X'FF' or X'E9'

  SINGLE or SINGLE with WAIT option
  any value other than X'FF' or X'E9'

Related concepts

Timeout intervals for IMS Connect acknowledgments to OTMA
You can specify a timeout interval that determines how long OTMA waits for an acknowledgment from IMS Connect. You can also specify a timeout tpipe queue to hold commit-then-send (CM0) output after the timeout interval has expired.

Timer interval specifications

You can specify timer values in several incremental ranges.

The values in each range are selected by entering a hexadecimal value in the IRM_TIMER field in the IRM of the input message from the client. The hexadecimal values that can be specified and the time intervals that they represent are:

- Increments of one one-hundredth of a second are represented by values of X'01' to X'19', as shown in Table 61 on page 313.
- Increments of five one-hundredths of a second are represented by values of X'1A' to X'27', as shown in Table 62 on page 314.
- Increments of one second are represented by values of X'28' to X'63', as shown in Table 63 on page 315.
- Increments of one minute are represented by values of X'63' to X'9E', as shown in Table 64 on page 316.
- Default timer values, the no-timer option, and the indefinite wait option, specified by X'00', X'E9', and X'FF' respectively, are shown in Table 65 on page 318.

The following table lists the IRM_TIMER values and their corresponding time in increments of one one-hundredth of a second.

Table 61. IRM_TIMER values in one one-hundredth of a second

Time	Hexadecimal value
.01 of a second	X'01'
.02 of a second	X'02'
.03 of a second	X'03'
.04 of a second	X'04'
.05 of a second	X'05'
.06 of a second	X'06'
.07 of a second	X'07'
.08 of a second	X'08'
.09 of a second	X'09'
.10 of a second	X'0A'
.11 of a second	X'0B'
.12 of a second	X'0C'
.13 of a second	X'0D'
.14 of a second	X'0E'
.15 of a second	X'0F'
.16 of a second	X'10'
.17 of a second	X'11'
.18 of a second	X'12'
.19 of a second	X'13'
.20 of a second	X'14'
.21 of a second	X'15'

Table 61. IRM_TIMER values in one one-hundredth of a second (continued)

Time	Hexadecimal value
.22 of a second	X'16'
.23 of a second	X'17'
.24 of a second	X'18'
.25 of a second	X'19'

The following table lists the IRM_TIMER values and their corresponding time in five one-hundredths of a second.

Table 62. IRM_TIMER values in five one-hundredths of a second

Time	Value
.30 of a second	X'1A'
.35 of a second	X'1B'
.40 of a second	X'1C'
.45 of a second	X'1D'
.50 of a second	X'1E'
.55 of a second	X'1F'
.60 of a second	X'20'
.65 of a second	X'21'
.70 of a second	X'22'
.75 of a second	X'23'
.80 of a second	X'24'
.85 of a second	X'25'
.90 of a second	X'26'
.95 of a second	X'27'

The following table lists the IRM_TIMER values for time increments of one second each.

Table 63. IRM_TIMER time values in seconds

Time	Value
1 second	X'28'
2 seconds	X'29'
3 seconds	X'2A'
4 seconds	X'2B'
5 seconds	X'2C'
6 seconds	X'2D'
7 seconds	X'2E'
8 seconds	X'2F'
9 seconds	X'30'
10 seconds	X'31'
11 seconds	X'32'
12 seconds	X'33'
13 seconds	X'34'
14 seconds	X'35'
15 seconds	X'36'
16 seconds	X'37'
17 seconds	X'38'
18 seconds	X'39'
19 seconds	X'3A'
20 seconds	X'3B'
21 seconds	X'3C'
22 seconds	X'3D'
23 seconds	X'3E'
24 seconds	X'3F'
25 seconds	X'40'
26 seconds	X'41'
27 seconds	X'42'
28 seconds	X'43'
29 seconds	X'44'
30 seconds	X'45'
31 seconds	X'46'
32 seconds	X'47'
33 seconds	X'48'
34 seconds	X'49'

Table 63. IRM_TIMER time values in seconds (continued)

Time	Value
35 seconds	X'4A'
36 seconds	X'4B'
37 seconds	X'4C'
38 seconds	X'4D'
39 seconds	X'4E'
40 seconds	X'4F'
41 seconds	X'50'
42 seconds	X'51'
43 seconds	X'52'
44 seconds	X'53'
45 seconds	X'54'
46 seconds	X'55'
47 seconds	X'56'
48 seconds	X'57'
49 seconds	X'58'
50 seconds	X'59'
51 seconds	X'5A'
52 seconds	X'5B'
53 seconds	X'5C'
54 seconds	X'5D'
55 seconds	X'5E'
56 seconds	X'5F'
57 seconds	X'60'
58 seconds	X'61'
59 seconds	X'62'
60 seconds	X'63'

The following table lists the IRM_TIMER values and their corresponding time increments of one minute each.

Table 64. IRM_TIMER time values in minutes

Time	Value
1 minute	X'63'
2 minutes	X'64'
3 minutes	X'65'
4 minutes	X'66'

Table 64. IRM_TIMER time values in minutes (continued)

Time	Value
5 minutes	X'67'
6 minutes	X'68'
7 minutes	X'69'
8 minutes	X'6A'
9 minutes	X'6B'
10 minutes	X'6C'
11 minutes	X'6D'
12 minutes	X'6E'
13 minutes	X'6F'
14 minutes	X'70'
15 minutes	X'71'
16 minutes	X'72'
17 minutes	X'73'
18 minutes	X'74'
19 minutes	X'75'
20 minutes	X'76'
21 minutes	X'77'
22 minutes	X'78'
23 minutes	X'79'
24 minutes	X'7A'
25 minutes	X'7B'
26 minutes	X'7C'
27 minutes	X'7D'
28 minutes	X'7E'
29 minutes	X'7F'
30 minutes	X'80'
31 minutes	X'81'
32 minutes	X'82'
33 minutes	X'83'
34 minutes	X'84'
35 minutes	X'85'
36 minutes	X'86'
37 minutes	X'87'
38 minutes	X'88'

Table 64. IRM_TIMER time values in minutes (continued)

Time	Value
39 minutes	X'89'
40 minutes	X'8A'
41 minutes	X'8B'
42 minutes	X'8C'
43 minutes	X'8D'
44 minutes	X'8E'
45 minutes	X'8F'
46 minutes	X'90'
47 minutes	X'91'
48 minutes	X'92'
49 minutes	X'93'
50 minutes	X'94'
51 minutes	X'95'
52 minutes	X'96'
53 minutes	X'97'
54 minutes	X'98'
55 minutes	X'99'
56 minutes	X'9A'
57 minutes	X'9B'
58 minutes	X'9C'
59 minutes	X'9D'
60 minutes	X'9E'

The following table lists additional options that you can specify in the IRM_TIMER field and the value you use to specify them.

Table 65. Additional IRM_TIMER options

Timer option	Value
Use default values. For RESUME TPIPE calls and associated ACK messages, the default is .25 seconds. For all other SENDs, the default is the configuration file TIMEOUT value.	X'00'
Do not wait.	X'E9' C'Z'
Wait indefinitely. This setting is intended to support the auto option of the asynchronous output function.	X'FF'

Canceling a message timer

User-written IMS Connect client applications can cancel the active message timer when waiting on output from the data store.

The cancel timer feature prevents IMS Connect clients that have specified a large timeout interval from being lost in the event that the data store does not send a reply. Without the cancel timer feature an IMS Connect ST0PCLNT command would have to be issued to clear the socket connection. When a request to cancel the timer is submitted, IMS Connect notifies the client.

The cancel timer feature is supported by user-written message exit routines and the sample user message exit routines HWSSMPL0 and HWSSMPL1.

Note that if a client is waiting in a CONN state after issuing a RESUME TPIPE call and the data store is closed by IMS or a ST0PDS command, then the client receives an RSM message with RC= X'2C' (or decimal 44).

A cancel timer request is specified by a C in the IRM_F4 field and can be submitted from either a single instance of a client or two instances of the client with the same client ID.

To submit a cancel timer request from a single client instance:

1. Issue SEND of ACK.
2. Set local timer.
3. Issue READ for Response. Timer pops rather than receipt of data.
4. Issue Disconnect.
5. Issue Connect.
6. Issue SEND with Cancel Timer set in the IRM.
7. Issue Read for Cancel Timer. The user message exit issues return code 8 with a reason code of X'3B' (or decimal 59) in the RSM.
8. Issue Disconnect.

To submit a cancel timer request from two instances of the same client with the same client ID:

1. From first client
   a. Issue SEND of ACK (with, for example, a client ID of ICON01).
   b. Issue READ for Response.
2. From second client instance
   a. Issue Connect.
   b. Issue SEND with Cancel Timer set in the IRM (with client ID of ICON01).
   c. Issue Read for Cancel Timer. The user message exit issues return code 8 with a reason code of X'3B' (or decimal 59) in the RSM.
   d. Issue Disconnect.
3. First client receives an RSM with a return code of X'2C' (or decimal 44) and a reason code set to the value of the timer value. This instance remains connected and is in RECV state.
   a. Issue Disconnect or continue processing.

Timeout intervals for IMS Connect acknowledgments to OTMA

You can specify a timeout interval that determines how long OTMA waits for an acknowledgment from IMS Connect. You can also specify a timeout tpipe queue to hold commit-then-send (CM0) output after the timeout interval has expired.

You can specify a timeout interval by using the ACKTO parameter in the DATASTORE configuration statement. The timeout value specified on the ACKTO parameter applies to acknowledgments sent to OTMA for the following message types:

- Transaction messages sent to a remote IMS system

- CM0 output
- Send-then-commit (CM1) output

For transaction messages that are sent to a remote IMS system, if the ACK timeout interval expires, OTMA reroutes the transaction message to the timeout queue. If OTMA receives an ACK response from the local IMS Connect after a transaction message has timed out, OTMA issues a NAK with X'2B' return code to the local IMS Connect.

For CM0 output, when the timeout interval expires, OTMA removes the output from the tpipe queue and reroutes the output to either:

- A specified reroute tpipe queue
- A specified timeout tpipe queue
- The default OTMA timeout tpipe queue DFS$$TOQ

You can specify the name of a timeout tpipe queue for CM0 output and for transaction messages that are sent to remote IMS systems on the CM0ATOQ parameter in both the HWS and DATASTORE configuration statements.

On the HWS configuration statement, the CM0ATOQ parameter defines a default timeout tpipe queue for all data store connections defined to an instance of IMS Connect. On the DATASTORE statement, the CM0ATOQ parameter defines the timeout tpipe queue to be used by only the defined data store connection. Specifications for CM0ATOQ on the DATASTORE statement override the specification for CM0ATOQ made on the HWS configuration statement.

CM1 output is not rerouted to a timeout queue, because OTMA discards CM1 output if the timeout interval expires.

When a timeout occurs, and the IMS application does not reply to the IOPCB or complete a message switch to another transaction, OTMA issues a DFS2082 message for both CM0 and CM1 input messages, regardless of the transaction response mode.

Related concepts

Timeout specifications on input messages
Each and every input message from the IMS Connect client can set a different timeout value in the IRM_TIMER field of the fixed portion of the IMS request message (IRM) header.

Timeout specifications for IMS-to-IMS connections

For IMS application programs that send transactions to a remote IMS system by way of OTMA, you can specify timeout values on the RMTIMSCON and DATASTORE configuration statements.

The IDLETO keyword in the RMTIMSCON configuration statement controls how long IMS Connect keeps open a persistent socket connection if no additional messages are received from IMS for the connection.

The ACKTO keyword on the DATASTORE configuration statement controls how long OTMA waits for an acknowledgement from IMS Connect before rerouting the output to the timeout tpipe queue and issuing an error message.

The timeout interval for acknowledgements can also be set in OTMA by either the TIMEOUT keyword of the **/START TMEMBER** command or the T/O parameter in the OTMA client descriptor in the DFSYDTx PROCLIB member.

Related reference

RMTIMSCON statement (System Definition)
OTMA client descriptor syntax and parameters (System Definition)
/START TMEM command (Commands)

IMS Connect transaction expiration support

IMS Connect can adjust the expiration time for IMS transactions to match the timeout value of the socket connection on which the transaction is submitted.

If an expiration time is specified in IMS, transactions can expire and be discarded if IMS does not process them before the expiration time is exceeded.

The transaction expiration time for a transaction is set in the definition of the transaction in IMS and is not specific to OTMA. In IMS, you can set or modify the expiration time for a transaction in the following ways:

- When a transaction is defined during IMS system definition, by specifying the **EXPRTIME** parameter in the TRANSACT stage-1 system definition macro.
- When a transaction is created during runtime by using dynamic resource definition, by specifying the **EXPRTIME** keyword on the **CREATE TRAN** type-2 command.
- For a transaction created by the Destination Creation exit routine (DFSINSX0), the exit routine can set the expiration time.
- For an existing transaction, by specifying the **EXPRTIME** keyword on the **UPDATE TRAN SET(EXPRTIME)** type-2 command. Dynamic resource definition does not need to be enabled.

A transaction expiration time set by IMS Connect overrides any transaction expiration time that is specified in the definition of the transaction in IMS.

The following IMS Connect user message exits support setting a transaction expiration time:

- HWSSMPL0
- HWSSMPL1
- HWSSOAP1

IMS Connect clients that use HWSSMPL0, HWSSMPL1, or a user-written exit routine can instruct IMS Connect to set the expiration time for transactions by specifying X'01' (IRM_F1_TRANEXP) in the user section of the IRM prefix. IMS Enterprise Suite SOAP Gateway clients can modify the HWSSOAP1 user message exit routine to set the IRM_F1 TRANEXP field.

To match the expiration time to the timeout value of the socket connection, IMS Connect calculates when the client socket connection will time out by reading the value of either the TIMEOUT parameter of the TCPIP configuration statement or the IRM_TIMER field in the fixed section of the IRM prefix of input messages. IMS Connect then places that time in the user data section of the OTMA message prefix in store clock (STCK) format and notifies OTMA of the new expiration time by setting fields OMHDRXP1 and OMHDRSXP in the state data section of the OTMA message prefix.

When transaction expiration is enabled for transactions submitted by IMS Connect, OTMA monitors the processing of the transaction and, if the transaction expiration time has passed, OTMA discards the transaction and returns a message to IMS Connect.

Depending on when the transaction times out, OTMA might return either a NAK response to IMS Connect or message DFS3688I.

For detailed information about when OTMA checks for expired transactions, see “Specifying an expiration time for transactions to OTMA” on page 803.

If IMS Connect does not set a transaction expiration time, the transaction can still expire if the EXPRTIME parameter was specified when the transaction was originally defined to IMS.

Related concepts

“Specifying an expiration time for transactions to OTMA” on page 803
You can specify an expiration time for a transaction to reduce processing costs by preventing IMS from processing transactions that the client can no longer use.

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

TRANSACT macro (System Definition)

Setting a transaction expiration time with IMS Connect

You can instruct IMS Connect to set an expiration time for transactions submitted to IMS that matches the timeout value of the socket connection on which the transaction is submitted.

For IMS Connect to set a transaction expiration time, a timeout value for the socket connection must first be specified in either the TIMEOUT parameter of the TCPIP configuration statement or the IRM_TIMER field in the fixed section of the IRM prefix.

You can instruct IMS connect to set a transaction expiration time for a transaction by using one of the following methods:

- Specify X'01' (IRM_F1_TRANEXP) in the IRM_F1 field of the user section in the IRM prefix of the input message.

  When IRM_F1 is set to X'01', the user message exit sets the OMHDRIST field to X'01' (OMHDRXP1) in the state data section of the OTMA header.
- Modify any of the following user message exit routines to set the IRM_F1 field to X'01' (IRM_F1_TRANEXP).
  - HWSSMPL0
  - HWSSMPL1
  - HWSSOAP1

  When IRM_F1 is set to X'01', the user message exit sets the OMHDRIST field to X'01' (OMHDRXP1) in the state data section of the OTMA header.

Related reference

"IRM structures for IMS Connect client messages" on page 213
IMS Connect expects all client messages that it receives to start with a four byte total length field, followed by an IMS request message (IRM) header, followed by the message data segments.

"OTMA state data fields used by IMS Connect" on page 257
The tables in this topic describe the fields of the OTMA state data header and the order of those fields.

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

Retrieval of output on OTMA tpipe hold queues

OTMA uses tpipe hold queues to send output to IMS Connect that must be queued for delivery. To retrieve output from the tpipe hold queue, the IMS Connect client issues a RESUME TPIPE request that specifies the name of the tpipe on which the output messages are queued.

The types of output messages that OTMA sends via the tpipe hold queue include:

- Synchronous callout messages that IMS application programs send by issuing the DL/I ICAL call.
- Output, such as asynchronous callout messages, that IMS application programs send by issuing the DL/I ISRT call to an alternate PCB (ALTPCB).
- CM0 IOPCB output messages for which the receiving OTMA client returned a NAK.
- Response messages to CM0 SendOnly input.

For CM0 transactions sent by a single client that uses a send and receive interaction, the name of the tpipe is typically the client ID. The output from the CM0 transaction can be retrieved in the following ways:

- By the original client with the client ID that matches the tpipe name.
- If the original client terminated, by another client that uses the same client ID that matches the tpipe name.
- By another client that specifies the tpipe name as an alternate client ID in the RESUME TPIPE request.

For synchronous callout messages, the tpipe name is typically defined in IMS on an OTMA destination descriptor. For asynchronous callout messages, the tpipe name can be defined by either an OTMA destination descriptor or an OTMA routing exit routine. For both types of callout messages, the tpipe

name is usually specified as an alternate client ID in the RESUME TPIPE request that retrieves the callout messages. The alternate client ID must match the tpipe name that is specified in the OTMA destination descriptor or in the OTMA routing exit routines.

Tpipe names are also specified as an alternate client ID when a tpipe supports parallel active RESUME TPIPE requests.

Note: Synchronous callout request messages are handled by OTMA and IMS Connect in much the same way as asynchronous output. That is, synchronous callout request messages are retrieved by issuing a RESUME TPIPE call. Many of the same rules and guidelines for retrieving asynchronous output also apply to retrieving synchronous callout request messages. However, for synchronous callout messages, if the RESUME TPIPE call from a client is connected to a different IMS Shared Queue member from the one that initiated and processed the synchronous callout request, the client will not receive the message. Because synchronous callout requests are queued to the tpipe hold queue, they are known only by the IMS that owns the tpipe. Super member function is honored for synchronous callout requests only when multiple IMS Connect clients are connected to the same IMS Shared Queue member.

IMS Connect communicates the presence of asynchronous output to the client from a CM0 output response message in one of the following ways:

- By returning the flag CSM_AMSG in the CSM_FLG1 field in the CSM (complete status message).
- By returning the flag RSM_AMSG in the RSM_FLG1 field in the RSM (request status message).

If you do not use IMS Connect to retrieve output from an OTMA tpipe hold queue, your client application does not need to analyze the CSM or the RSM. IMS Connect communicates the presence of asynchronous output regardless of whether a client application requests the asynchronous output.

Use the RESUME TPIPE call to retrieve the asynchronous output from the client. You can retrieve asynchronous output on both persistent and transaction sockets.

Restriction: The IMS TM Resource Adapter supports only the asynchronous option, SINGLE.

RESUME TPIPE/receive protocol

IMS Connect clients use the RESUME TPIPE protocol to retrieve commit-then-send (CM0) output or synchronous callout requests from a tpipe hold queue in IMS.

Synchronous callout requests are issued by IMS application programs running in an IMS dependent region. Callout requests are for data or services from a provider that is external to the IMS installation. In a callout scenario, the IMS application is the client and the external provider is the server.

Asynchronous output from IMS can include response messages from an IMS transaction and asynchronous callout messages.

For user-written IMS Connect client application programs, the RESUME TPIPE call is coded by specifying the following field values in the IRM prefix:

IRM_F4
R character value (IRM_F4_RESUMET).

IRM_F5
Hexadecimal value that specifies the retrieval option for the RESUME TPIPE call. The retrieval option specifies how many messages a single RESUME TPIPE call can retrieve and how long the IMS Connect client waits for additional messages when no messages remain on the tpipe hold queue.

IRM_F6
A value of X'80' (IRM_F6_NWSE) specifies that the RESUME TPIPE call can retrieve synchronous or asynchronous callout requests that contain the *NETSID* and *NETUID* distributed network security information segments.

IRM_TIMER
The timeout value for the RESUME TPIPE call. The value of IRM_TIMER determines how long the client waits for new messages if no output is currently on the tpipe hold queue.

When retrieving synchronous callout messages, in addition to the above fields, you must also specify the following fields to set the IRM architecture level (IRM_ARCH3) and specify whether the IMS Connect client can retrieve synchronous callout message only, or asynchronous output as well:

IRM_ARCH
X'03' (IRM_ARCH3). IRM_ARCH3 is required when retrieving synchronous callout messages.

IRM_F0
One of the following values must be specified when retrieving synchronous callout messages:

- X'80' (IRM_F0_SYNONLY). When IRM_F0_SYNONLY is specified, only synchronous callout messages are returned to the IMS Connect client.
- X'40' (IRM_F0_SYNASYN). When IRM_F0_SYNASYN is specified, both synchronous callout messages and asynchronous output are returned to the IMS Connect client.

Related reference

"Format of fixed portion of IRM in messages sent to IMS Connect" on page 214
The IMS request message (IRM) header contains a 28-byte fixed-format section that is common to all messages from all IMS Connect client applications that communicate with IMS TM.

"Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines" on page 218
Following the 4-byte length field and the 28-byte fixed portion of the IMS request message (IRM) header in IMS Connect client input messages, user-written client applications supported by HWSSMPL0, HWSSMPL1, or user-written message exits can include a user-defined section in the IRM.

"Timer interval specifications" on page 312
You can specify timer values in several incremental ranges.

Examples flows for the RESUME TPIPE protocol

The following figures show examples of the RESUME TPIPE protocol to receive asynchronous commit-then-send (CM0) output.

In the following figure, the CM0 flow enqueues IMS output before sending it to the client with the client application sending a positive acknowledgment (ACK) for both outputs. The ACK removes the output from the IMS queue.

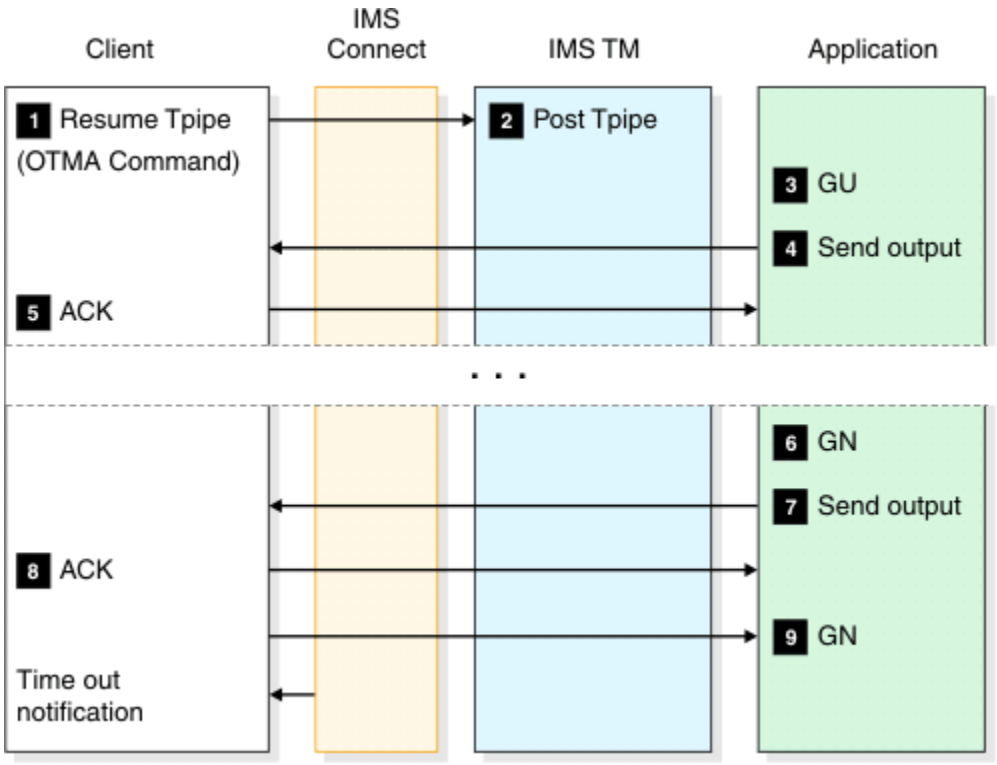


Figure 33. Commit-then-send, receive asynchronous output (client waits for output) flow

The sample flow in the previous figure assumes the following:

- The client sends the OTMA call RESUME TPIPE to ask IMS OTMA to post the named Tpipe (the client name).
- The client issues a RECEIVE request to receive the output from IMS.
- The client sends ACK to IMS (required for commit-then-send).
- The client receives the next output from IMS.
- The client sends ACK to IMS.
- The client waits for the next output from IMS, or for Time out notification.

In the following figure, the commit-then-send flow enqueues IMS output before sending it to the client and the client application sends a positive acknowledgment (ACK) for the first output (removing the output from the IMS queue) and a NAK to the second output (which results in the output remaining in the queue).

Requirement: Use this protocol with the timeout function. Otherwise, the client hangs if there are no more messages to send.

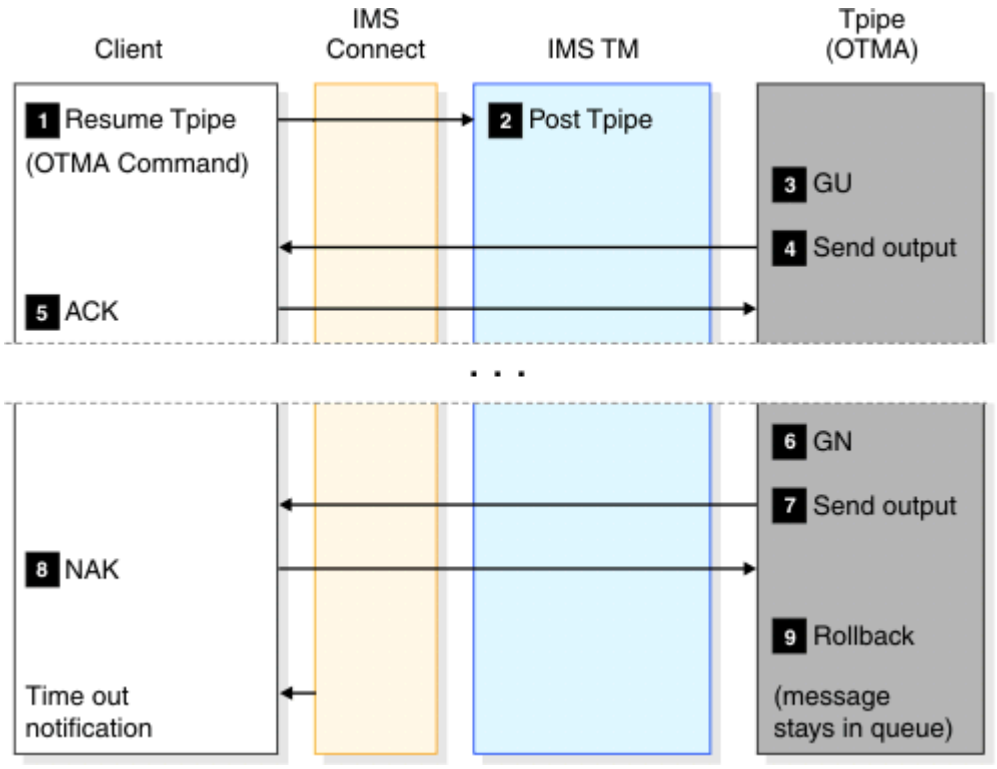


Figure 34. Commit-then-send, receive asynchronous output (output remains in queue) flow

The sample flow in the previous figure assumes the following:

- The client sends the OTMA call RESUME TPIPE to ask IMS OTMA to post the named Tpipe (the client name).
- The client receives the output from IMS.
- The client sends ACK to IMS (required for commit-then-send).
- The client receives the next output from IMS.
- The client sends NAK to IMS.
- The message stays in the queue.

Implementing asynchronous output support

You implement asynchronous output support by enabling the receipt of the asynchronous output.

The end user of the client application can decide when to request the asynchronous output, or the client application itself can decide when to request the asynchronous output.

Recommendation: Implement asynchronous output support so that the *end user*, not the client application, decides when to request the asynchronous output. Such an implementation provides these benefits:

- Ensures that the transaction input and output is separated from the asynchronous output.
- Enables the end user to select, at a time interval of their choice, when to retrieve the asynchronous output.

Regardless of whether or not the end user or the client application requests the asynchronous output, the following actions must occur, in this order:

1. Issue a CONNECT command.
2. A TCP/IP SEND of an OTMA RESUME TPIPE call, immediately followed by a TCP/IP READ function from the primary client application.

3. A TCP/IP SEND of an ACK or NAK response on the receipt of the output message. If the ACK was sent with a timer value of NOWAIT (NOWAIT is only valid for a RESUME TPIPE call with SINGLE or SINGLE with WAIT option), go to step 5. If NAK was sent, go to step 5.
4. A TCP/IP READ function from the primary client application. Repeat steps 2 and 3 until either all messages have been received, until the end user has received all of the messages that they want, until an error occurs, or until time out notification occurs.
5. Issue a DISCONNECT command, if you are using transaction sockets. If you are using persistent sockets, the connection is still connected.

Enabling end user asynchronous output requests

You can easily implement the CONNECT, RESUME TPIPE, READ, ACK/NAK, and DISCONNECT functions on the client application's screen with the buttons on the graphical user interface.

- Create a CONNECT button.
- Create a RESUME TPIPE button to send a RESUME TPIPE call to IMS Connect. IMS Connect will then send a RESUME TPIPE request to OTMA.
- Create a READ button to issue a TCP/IP READ request. OTMA will send a message to IMS Connect following the RESUME TPIPE or ACK response.
- Create an ACK/NAK button.
- You can also combine the READ and ACK requests into a single button that issues the READ request, then sends an ACK on receiving the message.
- Create a DISCONNECT button.

Retrieving output with parallel RESUME TPIPE requests

To increase the throughput of OTMA output messages, especially callout request messages, you can enable an OTMA tpipe to support multiple active **RESUME TPIPE** requests in parallel by specifying MULTIRTP=Y in either the HWS or the DATASTORE IMS Connect configuration statement.

To preserve the serial processing of output messages, you can also disable the parallel processing of **RESUME TPIPE** requests by specifying MULTIRTP=N.

By default, OTMA creates tpipes that support only a single active **RESUME TPIPE** request and queues any additional **RESUME TPIPE** requests until the active **RESUME TPIPE** request terminates. Supporting only a single active **RESUME TPIPE** request provides more control over the order in which the output messages from OTMA are processed.

Enabling support for the parallel processing of multiple active **RESUME TPIPE** requests can significantly increase the throughput of an OTMA tpipe for output messages, particularly those for synchronous or asynchronous callout requests, and can significantly improve failover protection for OTMA tpipes. Although OTMA sends the output messages in the order in which they are created from the IMS application programs, differences in the performance of both the network connections and the IMS Connect client application programs make predicting the order in which the output is acknowledged and processed unpredictable.

If the MULTIRTP parameter is specified on both the HWS statement and a DATASTORE statement, the specification on the DATASTORE statement overrides the specification on the HWS statement. If the MULTIRTP parameter is not specified on either the HWS or the DATASTORE statement, the parallel processing option is determined by the MULTIRTP value in effect in OTMA.

Related concepts

“OTMA tpipe support for parallel processing of multiple active RESUME TPIPE requests” on page 769

When MULTIRTP=Y is specified in an OTMA client descriptor, the OTMA tpipes that are associated with the OTMA client can support multiple active resume tpipe requests in parallel, unless the MULTIRTP specification is overridden by the client.

Specifying the IMS Connect default for parallel RESUME TPIPE request support

The specification of the MULTIRTP parameter in the HWS configuration statement defines the default option for the parallel processing of RESUME TPIPE requests for all data store connections from this IMS Connect instance.

The MULTIRTP value that is specified in the HWS statement can be overridden for individual data store connections by specifying the MULTIRTP parameter in the DATASTORE statement that defines the data store connection.

- To set the default processing option for RESUME TPIPE requests on all data store connections from this instance, specify one of the following MULTIRTP values:

  ␣
  If left blank, or the MULTIRTP parameter is omitted, specifies that, by default, data store connection requests from this IMS Connect instance do not include any specification for MULTIRTP support. MULTIRTP support is determined by either the IMS Connect data store definition or the OTMA client descriptor in the DFSYDT*x* member of the IMS PROCLIB data set.

  N
  Specifies that IMS Connect data store connection requests indicate that IMS Connect requires an OTMA TPIPE that supports only a single active resume TPIPE request. Output messages on the TPIPE are sent serially.

  Y
  Specifies that IMS Connect data store connections require an OTMA TPIPE that can support multiple active resume TPIPE requests in parallel. When a TPIPE has multiple active resume TPIPE requests from multiple data store connections, OTMA sends the callout or CM0 output messages on the first available data store connection that is in a receive state.

Specifying support for parallel RESUME TPIPE requests for a data store connection

You can enable or disable support for parallel RESUME TPIPE requests for an individual IMS Connect data store connection by specifying the MULTIRTP parameter in the definition of the data store connection.

The MULTIRTP specification in the definition of a data store connection overrides the default MULTIRTP value of the IMS Connect instance.

- To set the processing option for RESUME TPIPE requests for an individual data store connection, specify one of the following MULTIRTP values:

  N
  Specifies that this data store connection requires an OTMA tpipe that supports only a single active RESUME TPIPE request. Output messages on the TPIPE are sent serially.

  Y
  Specifies that this data store connection requires an OTMA tpipe that supports multiple active RESUME TPIPE requests in parallel. When a tpipe has multiple active RESUME TPIPE requests from multiple DATASTORE connections, OTMA sends the callout or CM0 output messages on the first available connection that is in a receive state.

  #
  Specifies that MULTIRTP support for this data store connection is determined by the MULTIRTP value in effect for OTMA.

Implementing parallel RESUME TPIPE requests

To retrieve output from an OTMA tpipe that supports multiple active **RESUME TPIPE** requests, each IMS Connect client that issues a RESUME TPIPE request specifies the name of the tpipe as an alternate client ID.

How you specify an alternate client ID differs depending on the type of IMS Connect client that you are using.

To retrieve output from an OTMA tpipe that supports parallel **RESUME TPIPE** requests:

- For user-provided IMS Connect clients, issue the **RESUME TPIPE** request with the following values specified in the IRM message header:

  X'01' in the **IRM_ARCH** field of the fixed IRM section
  The name of the target tpipe in the **IRM_RT_ALTCID** field of the user-defined section of the IRM. The name of the target tpipe is typically defined in IMS by the TPIPE parameter in an OTMA destination descriptor.
- For IMS TM resource adapter client applications, the client application uses the `altClientID` property of the `IMSInteractionSpec` object in either of the following programming models:

  Asynchronous output programming model
  Asynchronous callout programming model
  Synchronous callout programming model

Until the maximum number of active RESUME TPIPE requests for the tpipe is reached, each **RESUME TPIPE** request that connects to the tpipe is active and can receive output. The maximum number of active RESUME TPIPE requests for a tpipe is defined by the LIMITRTP parameter in the OTMA client descriptor for the client.

Resolving problems with parallel RESUME TPIPE requests

Diagnosing problems with communication between IMS Connect and an OTMA tpipe that supports parallel RESUME TPIPE requests is complicated by the fact that multiple RESUME TPIPE requests can be active on a tpipe at the same time.

Also, because OTMA identifies OTMA clients by the name of the tpipe that they use, OTMA knows only the tpipe name that is specified by the client in the alternate client ID field and not the actual client ID.

To help identify the actual client ID, when parallel RESUME TPIPEs are supported, you can use the RESUME TPIPE token, the alternate client ID, and the client ID to correlate an IMS Connect client with the specific RESUME TPIPE request it has active with the OTMA tpipe.

To identify which client issued each **RESUME TPIPE** request that is active on an OTMA tpipe that supports parallel RESUME TPIPE requests:

- Issue the IMS command **/DISPLAY TMEMBER** *ims_connect_name* TPIPE *tpipe_nm* OUTPUT
- Note the IMS Connect instance and the relevant RESUME TPIPE tokens that are displayed in the output of the **/DISPLAY TMEMBER** command.
- Issue the IMS Connect command **QUERY IMSCON TYPE(CLIENT)** RTTOKEN(*rt_token_number*) SHOW(ALTCID RTTOKEN)
- Locate the RESUME TPIPE token in the information that is displayed by the QUERY IMSCON command. The client ID that issued the RESUME TPIPE call is displayed under the heading ClientID on the same row as its associated token.

In the following example, CLIENT03 on the IMS Connect instance HWS1 issued a RESUME TPIPE call to OTMA tpipe CLIENT99. However, because OTMA recognizes OTMA clients by the name of the tpipe that they use, only CLIENT99 is displayed, and not CLIENT03.

To determine which specific RESUME TPIPE call was issued by CLIENT03, the IMS command /DISPLAY TMEMBER is issued with the HWS1 and TPIPE ALL specified to display the RESUME TPIPE tokens of the

RESUME TPIPE calls that are currently active on the tpipe. To limit the output to a specific tpipe, specify the tpipe name, CLIENT99 in this case, instead of the ALL keyword.

```
/DIS TMEM HWS1 TPIPE ALL OUTPUT
DFS000I     MEMBER/TPIPE          ENQCT      DEQCT        QCT      INPCT STATUS
DFS000I     HWS1
DFS000I    -CLIENT99                 0          0          0          0 PMRY
DFS000I    -CLIENT99                 0          0          0          0 HLDQ
DFS000I    -CLIENT99              RT CBB45E89AAF02E8A   OPT A     MODE S
```

After the RESUME TPIPE token is determined, the QUERY IMSCON TYPE(CLIENT) command is issued with the token specified on the RTTOKEN keyword and SHOW(ALTCID) specified:

```
QRY IMSCON TYPE(CLIENT) RTTOKEN(CBB45E89AAF02E8A) SHOW(ALTCID RTTOKEN)
```

The client ID that corresponds to the RTTOKEN and the ALTCID is displayed with the RTTOKEN and ALTCID values:

```
ClientID MbrName            CC ALTCID   RTToken
CLIENT03 IMSPLEX1            0 CLIENT99 CBB45E89AAF02E8A
```

Managing the retrieval of output messages

When retrieving either asynchronous output messages or synchronous callout request messages with the RESUME TPIPE call, you have options regarding how messages are returned.

You specify your retrieval options in the IRM of the RESUME TPIPE request message and the user message exit sets the options in the OTMA header.

The retrieval options that you can specify include:

- Single
- Single with wait
- Noauto
- Nooption
- Auto

The IMS Connect user message exits HWSSMPL1 and HWSSMPL0 support all these options. To choose a type of message control, the client code sets the IRM field IRM_FLG5 to be one of the following values:

IRM_F5_ONE
Retrieves a single message (single).

IRM_F5_SWAIT
Waits for a single message if none are currently present in the IMS message queue (single with wait).

IRM_F5_NOAUTO
Retrieves all messages that have been queued (noauto).

IRM_F5_AUTO
Retrieves all messages that have been queued, then retrieves any additional messages that are queued later (auto).

IRM_F5
Makes a RESUME TPIPE call function like NOAUTO (nooption) when set to X'00'.

The HWSSMPL0 and HWSSMPL1 user message exits default to the noauto type of asynchronous output message management.

The following subsections describe the asynchronous output message control options in detail.

Single message control

When using the single message control option (by setting field IRM_F5 to IRM_F5_ONE), the client can receive only a **single message**.

If there are no messages in the IMS OTMA Asynchronous Queue for the client ID when the request is made, no message will be returned and a time out will occur. Using the single message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
    a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE call processing.
    b. If the socket type is a transaction socket, the RESUME TPIPE call must be issued after the CONNECT function.
2. Client issues RESUME TPIPE call with the correct IRM settings.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK or NAK to IMS Connect.
    a. The ACK or NAK can be sent with a user-specified numeric timeout value.

       Or
    b. Specify NOWAIT for the timeout value.
5. If a numeric timeout value is specified, the client must issue a RECEIVE function to receive the timeout notification. If the NOWAIT option is specified, no timeout notification is sent. Therefore, the client must not issue a RECEIVE function if NOWAIT is specified.
6. IMS Connect disconnects the Socket from the Host end if the socket connection is a transaction socket. If the socket connection is a persistent socket, IMS Connect does not disconnect the socket.
7. Client must issue a DISCONNECT function if the socket connection is a transaction socket. If the socket is a persistent socket, the client can either DISCONNECT the socket or choose to send in a new request such as SENDONLY, SEND of transaction code and data, or issue another RESUME TPIPE call.

If the client responds with a NAK rather than an ACK, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, and can be re-retrieved later. IMS Connect will continue to process as described in events five through seven when a NAK is sent to IMS Connect by the Client.

Single with wait message control

When using the single with wait message control option (by setting IRM_F5 to IRM_F5_SWAIT), the client can receive only one single message; however, unlike single message control, the single with wait message control can receive a message that is placed in the IMS OTMA Asynchronous Queue for the client ID.

Using the single with wait message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
    a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE call processing.
    b. If the socket type is transaction socket, then the RESUME TPIPE call processing must be issued after the CONNECT function.
2. Client issues RESUME TPIPE call, with the correct IRM settings.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK or NAK to IMS Connect.
    a. The ACK or NAK can be sent with a user-specified timeout notification.

       Or
    b. Specify NOWAIT for the timeout value.

5. If a numeric timeout value is specified, the client must issue a RECEIVE function to receive the timeout notification. If the NOWAIT option is specified, no timeout notification is sent. Therefore, the client must not issue a RECEIVE function if NOWAIT is specified.
6. IMS Connect disconnects the Socket from the Host end if the socket connection is a transaction socket. If the socket connection is a persistent socket, IMS Connect does not disconnect the socket.
7. Client must issue a DISCONNECT function if the socket connection is a transaction socket. If the socket is a persistent socket, the client can either DISCONNECT the socket or choose to send in a new request such as SENDONLY, SEND of transaction code and data, or issue another RESUME TPIPE call.

If the client responds with a NAK rather than an ACK, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, and can be re-retrieved later. IMS Connect will continue to process as described in events five through seven when a NAK is sent to IMS Connect by the Client.

Noauto message control

When using the noauto message control option (by setting field IRM_F5 to IRM_F5_NOAUTO), the client can receive **all** of the messages on the OTMA Asynchronous Queue.

Using the noauto message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
   a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE call processing.
   b. If the socket type is transaction socket, then the RESUME TPIPE call processing must be issued after the CONNECT function.
2. Client issues RESUME TPIPE call.
3. Client issues RECEIVE function to receive the asynchronous output.
4. Client sends ACK message to IMS Connect.
5. Client repeats events three and four until event six occurs.
6. IMS Connect disconnects the socket from the host end, unless the socket is persistent. If the socket is persistent, IMS Connect sends a timeout message and the socket remains connected, which allows the client to either disconnect or send another request.
7. Client issues DISCONNECT function.

Using the noauto message control option, the client can always terminate by issuing a DISCONNECT function after sending an ACK message to IMS Connect.

If the client responds with a NAK message rather than an ACK, the message that received the NAK response is put back on the OTMA asynchronous hold queue and can be retrieved later. IMS Connect terminates the socket as described in event six, unless the socket is persistent.

Nooption message control

When using the nooption message control option (by setting field IRM_F5 to X'00'), the client can receive **all** of the messages on the OTMA Asynchronous Queue.

Using the nooption message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
   a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE call processing.
   b. If the socket type is transaction socket, then the RESUME TPIPE call processing must be issued after the CONNECT function.
2. Client issues RESUME TPIPE call.
3. Client issues RECEIVE function to receive the asynchronous output.
4. Client sends ACK message to IMS Connect.

5. Client repeats events three and four until event six occurs.
6. IMS Connect disconnects the socket from the host end, unless the socket is persistent. If the socket is persistent, IMS Connect sends a timeout message and the socket remains connected, which allows the client to either disconnect or send another request.
7. Client issues DISCONNECT function.

Using the nooption message control option, the client can always terminate by issuing a DISCONNECT function after sending an ACK to IMS Connect.

If the client responds with a NAK message rather than an ACK, the message that received the NAK response is put back on the OTMA asynchronous hold queue, and can be retrieved later. IMS Connect terminates the socket as described in event six above, unless the socket is persistent.

Auto message control

When using the auto message control option (by setting field IRM_F5 to IRM_F5_AUTO), the client can receive all of the messages on the OTMA Asynchronous Queue, and any messages that are placed on the OTMA Asynchronous Queue after the current messages are all removed.

Using the auto message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
2. Client issues RESUME TPIPE call.
3. Client issues RECEIVE function to receive the asynchronous output.
4. Client sends ACK message to IMS Connect.
5. Client repeats events three and four.

If all messages have been removed from the queue, event three will remain active (that is, in receive state) until the user-specified timer supplied in the IRM has expired. IMS Connect will then terminate the socket, unless the socket is persistent.

Recommendation: If event three or event five receives a disconnect of the socket, the client should disconnect and then wait for a time interval before repeating events one through five.

Using the auto message control option, the client can always terminate a non-persistent socket connection in either one of the following ways:

- Respond to the output message with a NAK response
- Send a DEALLOCATE request rather than an ACK response

The message being processed is put back on the IMS output queue, and IMS Connect terminates the socket, unless the socket is persistent.

If the client responds with an ACK message, then issues a DISCONNECT, the connection is only terminated between the client and TCP/IP; the client remains in a CONN state with IMS Connect. When IMS Connect attempts to send the next asynchronous output message, IMS Connect is notified that the connection has been lost. IMS Connect does not acknowledge (NAK) OTMA, and the message is put back on the IMS output queue. IMS Connect then terminates the socket. If the client issues an ACK message and then issues a DISCONNECT, followed by a connect and transmittal of data, IMS Connect responds with a duplicate client ID error and disconnects the socket connection.

If the client responds with a NAK message rather than an ACK in events three or five, the message that received the NAK message is put back on the OTMA asynchronous hold queue, IMS Connect terminates the socket, and the messages can be retrieved later.

Note: The IMS Connect AUTO support is based on the premise that the socket connection is dedicated as an output-only device. Combining RESUME TPIPE calls (with auto asynch option specified) with transactions on the same socket connection or SENDONLY on a persistent socket, can yield unpredictable results. If you want to change from RESUME TPIPE auto option mode to a mode that will allow for transaction processing, you must change the auto asynch option by performing one of the following options:

1. NAK one of the RESUME TPIPE outputs. This will change the asynch mode from auto to noauto. To return to auto mode, a RESUME TPIPE call with auto must be specified.
2. On a timeout notification associated with the RESUME TPIPE AUTO, the client application can disconnect, reconnect, and issue a RESUME TPIPE call with single, single with wait, noauto, or nooption with a very short IRM_TIMER value. The IRM_TIMER value should be small, so that a timeout notification can be returned immediately. Issuing a RESUME TPIPE call with one of the four asynch mode options, changes the mode from auto to one of the specified options. After the RESUME TPIPE call is issued and a timeout notification is returned, the client application can send in a transaction.
3. On a timeout notification associated with the RESUME TPIPE AUTO, the client application can disconnect, reconnect, and issue a RESUME TPIPE call with single, single with wait, noauto, or nooption with any valid IRM_TIMER value. Upon receiving an output message, send an ACK with IRM_TIMER set to nowait or a valid value. If the IRM_TIMER value is set to nowait, the client can then send in a transaction. If the IRM_TIMER is set to a valid value, after receiving the timeout notification, the client application can then send in a transaction.

Related concepts

"Timeout specifications on input messages" on page 311
Each and every input message from the IMS Connect client can set a different timeout value in the IRM_TIMER field of the fixed portion of the IMS request message (IRM) header.

Execution time out during RESUME TPIPE call processing with auto message control option

If you are using RESUME TPIPE calls with the auto message control option and the IRM_TIMER value times out, you might experience some unpredictable results.

If the auto option is selected on the RESUME TPIPE call and a timeout occurs, to get the timeout notification and send transactions again, you must change the auto option processing mode to noauto. To get out of the auto option processing mode, you can choose one of the following options:

- Issue a RESUME TPIPE call with the auto option and set a large IRM_TIMER value to ensure that the client application will NAK the output. When the output is NAK, OTMA will change the asynchronous mode from auto to noauto to stop the sending of asynchronous output. The client application then issues READ to retrieve the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue a RESUME TPIPE call with the noauto option and set any value in the IRM_TIMER field. After receiving ACK output, repeat READ of asynch output and SEND of ACK until a timeout notification is received. (Issuing a RESUME TPIPE call with noauto changed the processing mode from auto to noauto. This also reset the asynchronous mode in OTMA to noauto where OTMA no longer supports the automatic sending of asynch output when the IMS Message Queue is empty.) The client application then issues READ to retrieve the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue a RESUME TPIPE call with noauto option and set any value in the IRM_TIMER field. If you receive NAK output, the processing mode and OTMA Asynch mode is reset to noauto. Resetting the OTMA Asynch mode to noauto stops the sending of asynch output and the NAK output terminates the process. The client application then issues READ to retrieve the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue a RESUME TPIPE call with single option and set any value in the IRM_TIMER field. The OTMA Asynch mode is reset from auto to single and no more asynchronous messages are sent. After you receive ACK or NAK output with an IRM_TIMER setting that is anything other than NO_WAIT, the single option has been completed and the client application can issue a READ to get the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue a RESUME TPIPE call with single option and set any value in the IRM_TIMER field. The OTMA Asynch mode is reset from auto to single and no more asynchronous messages are sent. After you receive ACK or NAK output with an IRM_TIMER setting of NO_WAIT, the single option has been completed and the client application does not have to issue a READ to get timeout notification. The client application can start sending transactions to IMS Connect.

Values for asynchronous output processing

To retrieve asynchronous output from the OTMA tpipe hold queue, the IMS Connect client application must issue a RESUME TPIPE request. To issue a RESUME TPIPE call, you need to specify the socket type, commit mode, sync level, timer setting, and RESUME TPIPE call options.

To issue a RESUME TPIPE call, specify the following values:

- Socket Type
  Transaction or Persistent
- Commit Mode
  Zero
- Sync level
  Confirm
- Timer setting
  The timeout range required by your enterprise.
- RESUME TPIPE options
  Single, single with wait, auto, noauto, or nooption.

For example, if you want to create a dedicated output client that only receives unsolicited output, start a client application to complete the following sequence:

1. The client application performs a connection sequence.
2. The client application sends a RESUME TPIPE call with the correct settings in the IRM.

   Recommendation: Set the IRM_TIMER value to X'FF', which causes IMS Connect to override the TIMEOUT value in the configuration file and wait forever.
3. The client application sends a TCP/IP READ to receive the output message.
4. The client application sends an acknowledgment (ACK–Set the IRM_TIMER value to the same value you set on the RESUME TPIPE call.) and returns to the TCP/IP READ.

The timer interval that is set in IRM_TIMER is a different timer value from the one that is set in the IMS Connect configuration file (that value is TIMEOUT=).

The IRM_TIMER value is the wait value to wait for a RECEIVE issued from the client following a RESUME TPIPE call, or an ACK to the RECEIVEs following the RESUME TPIPE call.

Retrieving output from alternate OTMA tpipe hold queues

Client applications can retrieve the asynchronous output or callout messages from an alternate tpipe hold queue by specifying the name of the alternate tpipe as an alternate client ID a RESUME TPIPE call.

When IMS Connect passes a RESUME TPIPE call that specifies an alternate client ID to OTMA, OTMA returns to the caller any messages that are queued to the tpipe that matches the alternate client ID.

Specifying an alternate client ID is used to retrieve output in the following scenarios:

- By IMS TM resource adapter when client IDs are unknown because they are automatically generated at runtime.
- In Sysplex Distributor environments, where client applications typically do not know on which tpipe hold queue their output is queued.
- When a tpipe supports multiple active RESUME TPIPE requests, all of the clients that retrieve output from the tpipe specify the tpipe name as an alternate client ID.
- In callout environments, where the tpipe name is usually defined in IMS by an OTMA destination descriptor or, for asynchronous callout only, an OTMA routing exit. In this case, the callout messages are retrieved by specifying the tpipe name as an alternate client ID.

By using an alternate client ID, the client application programs can retrieve the output through any instance of IMS Connect by specifying the tpipe name in the alternate client ID field of either the IRM or the OTMA header.

When retrieving output by using an alternate client ID, a /DISPLAY TMEMBER TPIPE command displays the alternate client ID instead of the actual ID of the client that submitted the RESUME TPIPE request. Consequently, identifying which client submitted a particular RESUME TPIPE request can be a challenge if you need to diagnose a problem.

However, if the target tpipe supports parallel RESUME TPIPE requests, /DISPLAY TMEMBER TPIPE displays the RESUME TPIPE token, which can then be used to identify the true client ID in the output displayed by the IMS Connect command QUERY IMSCON TYPE(CLIENT).

- User-provided client applications can retrieve asynchronous output from an alternate tpipe by specifying the following in the IRM of the RESUME TPIPE call:

  X'01' in the IRM_ARCH field of the fixed IRM section
  The name of the target tpipe in the IRM_RT_ALTCID field of the user-defined section of the IRM
- For IMS TM resource adapter client applications, to retrieve asynchronous from an alternate tpipe the client application must specify the following values in the OTMA header:

  The name of the target tpipe in the OMUSR_RT_ALTCID field of the OTMA header
  OMUSR_AL02 in the OMUSR_ARCLEV field

Defining groups for shared asynchronous output

You can define groups of data store connections, IMS Connect instances, or both, in which the members of the group can retrieve the asynchronous output for any other member of the group. These groups use the OTMA super member function.

When a super member group is not defined, asynchronous output can be retrieved only by using the client ID of the originating client and the originating IMS Connect instance. The dependency on the original client ID and IMS Connect instance can present problems if, for example, either the originating client ID is unknown, as might be the case when the ID is automatically generated, or the original instance of IMS Connect is unknown, as might be the case when a product like the z/OS Sysplex Distributor is used.

Even when the client ID and IMS Connect instance are known, there is still a risk presented by the IMS Connect instance being a single point of failure; if the IMS Connect instance should fail, the asynchronous output could not be retrieved until the IMS Connect instance was brought back up.

You create a super member group by defining IMS Connect instances and individual data store connections as participants of the super member group. Each participant within the same super member group must specify the same super member name on the SMEMBER parameter in the IMS Connect configuration member in the IMS.PROCLIB data set. For IMS Connect, the SMEMBER parameter is on the HWS statement. For data store connections, the SMEMBER parameter is on the DATASTORE statement.

By default, the data store connections defined to an instance of IMS Connect are participants in the super member group that the instance is a participant in, if any; however, by specifying the SMEMBER parameter on the DATASTORE statement, a data store connection can belong to a different super member group or to no super member group at all. Specifications for super member groups made on the DATASTORE statement, override any super member group specification made on the HWS statement.

Note that if a DATASTORE statement does include the SMEMBER parameter but the HWS statement does, when the attributes of the data store connection are displayed they will not include the super member group name that, by default, the data store is a participant in. You must display the attributes of the HWS statement to see the super member group name.

- To define an IMS Connect instance as a participant in a super member group, specify a super member name on the SMEMBER parameter on the HWS configuration statement.

  The super member name specified on the HWS configuration statement becomes the default super member name for all data store connections defined to the instance of IMS Connect that do not specify a super member group of their own.

- To define a data store connection as a participant in a super member group other than the super member group specified on the HWS configuration statement, specify a super member name on the SMEMBER parameter on the DATASTORE configuration statement.

  A super member name specified on the DATASTORE statement overrides for this data store connection any super member name that is defined on the HWS configuration statement
- To prevent a data store connection from participating in a default super member group defined by an IMS Connect instance, specify SMEMBER='*####*' on the DATASTORE configuration statement that defines the data store connection.

Example

The following example of an IMS Connect configuration statement includes specifications for super member groups. The SMEMBER parameter on the HWS configuration statement defines the IMS Connect instance HWS1 as participating in the super member group SHRD and sets SHRD as the default super member group for all data store connections defined in this configuration member.

In the example, the data stores defined to the IMS Connect instance HWS1 each participate in super member groups as follows:

- IMS1 participates in the SHRD super member group defined on the HWS configuration statement because its DATASTORE statement does not include the SMEMBER parameter.
- IMSY and IMSN both participate in the super member group SHR1
- IMSX participates in super member SHR2, possibly with other data store connections from another instance of IMS Connect
- IMSA does not participate in any super member group

  Note: AT-TLS is the recommended method to enable SSL for IMS Connect.

```
HWS=(ID=HWS1,XIBAREA=100,RACF=Y,SMEMBER=SHRD)
TCPIP=(HOSTNAME=TCPIP,PORTID=(9999,LOCAL),RACFID=GOFISHIN,TIMEOUT=0,
IPV6=N,
EXIT=(HWSSMPL0,HWSSMPL1,HWSCSLO0,HWSCSLO1))
DATASTORE=(ID=IMS1,GROUP=XCFGRP1,MEMBER=HWS1,TMEMBER=IMS1)
DATASTORE=(ID=IMSY,GROUP=XCFGRP1,MEMBER=HWSY,TMEMBER=IMS1,SMEMBER=SHR1)
DATASTORE=(ID=IMSN,GROUP=XCFGRP1,MEMBER=HWSN,TMEMBER=IMS1,SMEMBER=SHR1)
DATASTORE=(ID=IMSX,GROUP=XCFGRP1,MEMBER=HWSX,TMEMBER=IMS1,SMEMBER=SHR2)
DATASTORE=(ID=IMSA,GROUP=XCFGRP1,MEMBER=HWSA,TMEMBER=IMSA,SMEMBER=####)
IMSPLEX=(MEMBER=HWSPLEX1,TMEMBER=PLEX1)
IMSPLEX=(MEMBER=HWSPLEX2,TMEMBER=PLEX2)
```

Related tasks

“Sharing asynchronous commit-then-send output: the OTMA super member function” on page 847
Hold-queue-capable OTMA clients, such as IMS Connect, can share asynchronous commit-then-send (CM0) output messages by enabling the OTMA super member function. The OTMA super member function is specifically designed to support multiple instances of IMS Connect in a z/OS Sysplex Distributor environment.

Related reference

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

Asynchronous output message flow

Implementing asynchronous output support forces a commit-then-send (commit mode 0) message flow. This flow requires an acknowledgment (ACK/NAK) from the client.

If an IMS transaction running in commit-then-send message flow sends a message to the client, and that message cannot be delivered, OTMA will react as though a NAK had been sent to OTMA from IMS Connect, and the message will be placed on the OTMA Hold Queue. OTMA will behave in this manner for whatever reason that the NAK gets sent (for example, because the z/OS cross-system coupling facility connection is not available, because IMS Connect has terminated, or because IMS Connect has lost communications with TCP/IP).

Related tasks

"Purging undeliverable commit-then-send output" on page 291
You can configure OTMA to purge commit-then-send (commit mode 0) IOPCB output when the output cannot be returned to the OTMA client application that initiated the transaction.

Related reference

"Output message from message exit to client" on page 239
Depending on the type of IMS Connect client and the user message exit used to support the client, the format of the message structure differs.

"OTMA header fields used by IMS Connect" on page 251
IMS Connect uses fields in the headers of messages sent to OTMA to communicate processing options and other information to IMS.

IMS Connect client call flows

The following examples show flows for IMS Connect client conversational and non-conversational transactions.

All sample flows shown apply to both persistent and transaction TCP/IP sockets, and all flows use this protocol: commit mode 1 (send-then-commit), synch level = confirm, with ACK and NAK.

The following sample flows are illustrated:

- Non-conversational, running to successful completion using ACK
- Conversational, running to successful completion using ACKs
- Non-conversational, where client sends NAK in response to message
- Conversational, where client sends NAK in response to one of the messages
- Non-conversational, terminated by Host application before successful completion of transaction
- Conversation terminated by Host application before successful completion of transaction

Non-conversational, running to successful completion using ACK

The following example shows a non-conversational flow with commit mode=1, synch level=confirm, and ACK (transaction runs to successful completion).

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST

  SEND---------------->IRM/TRAN/DATA ----------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/ACK-------------->RECEIVE

  RECEIVE<------------------RSM<----------------SEND DEALLOCATE CONFIRM
       RSM reason code = DEALLOCATE CONFIRM  X'61' (97)
     (97 = IMS Host application has committed the transaction)
```

Conversational, running to successful completion using ACKs

The following example shows a conversational flow with commit mode=1, synch level=confirm, and ACK (transaction runs to successful completion).

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST

  SEND---------------->IRM/TRAN/DATA ----------->RECEIVE
  RECEIVE<-------------- DATA/CSM<--------------SEND
  SEND-------------------->IRM/ACK-------------->RECEIVE
  SEND-------------------->IRM/DATA------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/ACK-------------->RECEIVE
                              .
                              .
                              .
```

```
  SEND------------------->IRM/DATA ------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND

  SEND------------------->IRM/ACK--------------->RECEIVE

  RECEIVE<-----------------RSM<-----------------SEND DEALLOCATE CONFIRM
         RSM reason code = DEALLOCATE CONFIRM  X'61' (97)
     (97 = IMS Host application has committed the transaction)
```

Non-conversational, where client sends NAK in response to message

The following example shows a non-conversational flow with commit mode=1, synch level=confirm, and NAK (transaction terminates with a NAK from client application).

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST

  SEND---------------->IRM/TRAN/DATA ----------->RECEIVE
  RECEIVE<---------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/NAK-------------->RECEIVE

  RECEIVE<---------------------------------- IMS MESSAGE "DFS555.."
```

Conversational, where client sends NAK in response to one of the messages

The following example shows a conversational flow with commit mode=1, synch level=confirm, and NAK (transaction terminates with a NAK from client application).

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST

  SEND---------------->IRM/TRAN/DATA ----------->RECEIVE
  RECEIVE<-------------- DATA/CSM<---------------SEND
  SEND-------------------->IRM/ACK--------------->RECEIVE
  SEND-------------------->IRM/DATA-------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/ACK--------------->RECEIVE
                              .
                              .
                              .
  SEND-------------------->IRM/DATA-------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/NAK--------------->RECEIVE

  RECEIVE<---------------------------------- IMS MESSAGE "DFS555.."
```

Non-conversational, terminated by Host application before successful completion of transaction

The following example shows a non-conversational flow with commit mode=1, synch level=confirm, and ACK (transaction terminated by host application before successful completion).

```
 CLIENT                  FLOW              IMS CONNECT
 REQUEST                                     REQUEST

  SEND---------------->IRM/TRAN/DATA ----------->RECEIVE
           Host Application abnormally terminates
  RECEIVE<---------------------------------- IMS MESSAGE "DFS555.."
```

Conversation terminated by Host application before successful completion of transaction

The following example shows a conversational flow with commit mode=1, synch level=confirm, and NAK (transaction terminated by host application).

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST
```

```
SEND---------------->IRM/TRAN/DATA ----------->RECEIVE
RECEIVE<---------------DATA/CSM<--------------SEND
SEND------------------>IRM/ACK-------------->RECEIVE

SEND------------------>IRM/DATA------------->RECEIVE
RECEIVE<---------------DATA/CSM<--------------SEND
SEND------------------>IRM/ACK-------------->RECEIVE
                            .
                            .
                            .
SEND------------------>IRM/DATA------------->RECEIVE
RECEIVE<---------------DATA/CSM<--------------SEND
SEND------------------>IRM/ACK-------------->RECEIVE

          Host Application abnormally terminates

RECEIVE<---------------------------------- IMS MESSAGE "DFS555.."
```

IMS Connect client message protocol sequence for IMS DFS messages and IMS command output

Table 66 on page 340 and Table 67 on page 341 show the required actions to be taken when different IMS DFS*nnnnn* messages or IMS command output is sent to the IMS Connect client. The two tables illustrate whether or not an ACK is required to be sent when the client receives an IMS DFS message or output from an IMS command, both for synch level Confirm and synch level None and for commit mode 0 (CM0) and commit mode 1 (CM1).

Note: The client code can test the CSM_FLG1 byte for the presence of the CSM_ACK_NAK flag; it can also test the RSM_FLG1 byte for the presence of the RSM_ACK_NAK flag. It performs this test to determine if an ACK or NAK is required. Otherwise, it performs the analysis outlined in Table 66 on page 340 and Table 67 on page 341.

Table 66 on page 340 and Table 67 on page 341 also define whether or not the client requires a READ in order to receive the "Deallocate Abort" response (RSM) from IMS Connect. Notes for both tables immediately follow Table 67 on page 341.

Table 66. Persistent sockets: client message protocol sequence for IMS DFS messages and IMS command output

Message output to client	CM1, Synch level confirm	CM1, Synch level none	CM0, Synch level confirm	CM0, Synch level none
Invalid transaction code DFS064	DFS064[1]	DFS064[1]	N/A	N/A
Transaction stopped DFS065	DFS065[1]	DFS065[1]	N/A	N/A
Transaction abended DFS555	DFS555[7]	DFS555[1]	N/A	N/A
Output DFS2082	DFS2082[2]	DFS2082[1]	N/A	N/A
IMS Command Output	Cmd output[1]	Cmd output[1]	N/A	N/A
Security Failure DFS1292	DFS1292[1]	DFS1292[1]	N/A	N/A
Segment greater than 32 K	DFS1294[5]	DFS1294[5]	N/A	N/A

Table 67. Transaction sockets: client message protocol sequence for IMS DFS messages and IMS command output

Message output to client	CM1, Synch level confirm	CM1, Synch level none	CM0, Synch level confirm	CM0, Synch level none
Invalid transaction code DFS064	DFS064[1]	DFS064[1]	DFS064[1]	N/A
Transaction stopped DFS065	DFS065[1]	DFS065[1]	DFS065[1]	N/A
Transaction abended DFS555	DFS555[7]	DFS555[1]	DFS555[7]	N/A
Output DFS2082	DFS2082[2]	DFS2082[1]	No output[3]	N/A
IMS Command Output	Cmd output[1]	Cmd output[1]	Cmd output[4]	N/A
Security Failure DFS1292	DFS1292[1]	DFS1292[1]	DFS1292[1]	N/A
Segment greater than 32 K	DFS1294[5]	DFS1294[5]	DFS1297[6]	N/A

Notes:

1. Does not require an ACK to DFS messages.
2. Requires both an ACK to DFS messages and a second read to get a deallocate response.
3. The read to receive the transaction output will time out. No data will be received. OTMA treats commit mode=0 and Synch level=Confirm as asynchronous output. If the IMS Host application does not return a message (ISRT to IOPCB), OTMA does not send a deallocate. The TIMEOUT= value specified in the IMS Connect configuration file will have to expire before the disconnect is complete.
4. Requires an ACK to command output. A second read is not required to get a deallocate response. The command output gets treated as asynchronous output.
5. Does not require ACK to DFS1294 output. A second receive is required to receive the DFS555 message.
6. Client will receive DFS1297 rather than DFS1294. The DFS1294 message does not require an ACK. No DFS555 message gets sent, so a second receive is not required. The application is committed, and the application output gets discarded because the segment is larger than 32 K.
7. Does not require an ACK to DFS messages.

Reason codes for commit mode=1, synch level=confirm

For CM1, there are three reason codes associated with a zero (0) return code, and two reason codes associated with an X'04' return code, which provide information to the client application. The sample flows illustrate how each of these codes are used. The code meanings are listed in the following table.

Table 68. Information reason codes for CM1, synch level=confirm

Return code	Reason code	Description
X'00'	94	Response - only output from host from non-conversation
X'00'	95	Conversation - last output from host from on conversation
X'00'	96	Conversation/response - middle of conversation

Table 68. Information reason codes for CM1, synch level=confirm (continued)

Return code	Reason code	Description
X'04'	97	Deallocate commit - successful completion of host application
X'04'	98	Deallocate abort - abnormal termination of host application

IMS Connect dead letter queue (HWS$DLQ)

In certain instances, if OTMA receives a NAK response from IMS Connect, OTMA stores the undelivered message on the IMS Connect dead letter queue. The IMS Connect dead letter queue is identified by the tpipe name HWS$DLQ.

The instances in which OTMA stores messages on HWS$DLQ include:

- When IMS Connect returns a NAK response to OTMA for a message that is missing the user data section of the OTMA header. In this case, IMS issues message HWSD0255W.
- When IMS Connect returns a NAK response to OTMA because IMS Connect could not process an asynchronous callout request. In this case, IMS issues message HWSP1510E.

Except in the case where a message has no user data section, you can retrieve messages on HWS$DLQ by specifying HWS$DLQ as the alternate client ID on the RESUME TPIPE call. Messages on HWS$DLQ that are missing the user data section of the OTMA header must be dequeued before subsequent messages on the queue can be retrieved.

To dequeue a message from HWS$DLQ:

1. Stop HWS$DLQ by issuing the command `/STOP TMEMBER` *`tmembername`* `TPIPE` *`hws$dlq`*.
2. Dequeue the message by issuing the command `/DEQUEUE TMEMBER` *`tmembername`* `TPIPE` *`hws$dlq`* `PURGE1`.
3. Start HWS$DLQ by issuing the command `/START TMEMBER` *`tmembername`* `TPIPE` *`hws$dlq`*.

To view the queue counts for HWS$DLQ issue the command `/DISPLAY TMEMBER` *`tmembername`* `TPIPE` *`hws$dlq`*.

Ping support for IMS Connect

To determine whether or not IMS Connect is available, you can send a ping request to IMS Connect. The ping support operates like a transaction and has the appearance of a transaction code and data. When you send the request `PING IMS_CONNECT`, the response is `PING RESPONSE`.

The user message exits, HWSJAVA, HWSSMPL1, HWSSMPL0, and HWSSOAP1 provide ping support. User message exits HWSCSLO0 and HWSCSLO1 do not support the ping function. If you write your own user message exit, you can choose to add the ping function support in your exit.

The communication sequence of the ping request consists of four steps. If you are using HWSSOAP1 user message exit, the third and the fourth steps, receive and disconnect, differ from the receive and disconnect steps for the HWSJAVA0, HWSSMPL0, and HWSSMPL1 user message exits.

For the HWSJAVA0, HWSSMPL0, and HWSSMPL1 user message exits, the communication sequence of the ping request is:

1. Connect.
2. Send `PING IMS_CONNECT` (must be sent in uppercase).
3. Receive:
   - If you are using HWSSMPL0 or HWSSMPL1, the user-written client application receives `HWSC0030I *PING RESPONSE* *CSMOKY*`.

- If you are using HWSJAVA0, the IMS TM Resource Adapter application receives `HWSC0030I *PING RESPONSE*` in the application data portion of the OTMA header and return code of 48 and reason code ICONSUCC in the user data portion of the OTMA header.

4. Resolve socket connection.

   IMS Connect maintains the socket connection, except in the following circumstances:

   - If the socket is not persistent, IMS Connect disconnects the socket.
   - If IMS Connect expects an ACK or NAK response from the client application, IMS Connect disconnects the socket.
   - If the client application is in IMS conversational mode, IMS Connect disconnects the socket.

For the HWSSOAP1 user message exit, the communication sequence of the ping request is:

1. Connect.
2. Send `PING IMS_CONNECT` (must be sent in uppercase).
3. Receive `PING_RESPONSE`.
4. Disconnect.

Related reference

“Format of the PING response” on page 244
The PING response is sent to IMS Connect clients after you send `PING IMS_CONNECT`, a ping request, to IMS Connect.

IMS TM Resource Adapter user message exit routine (HWSJAVA0) (Exit Routines)

User message exit routines HWSSMPL0 and HWSSMPL1 (Exit Routines)

SOAP Gateway exit routine (HWSSOAP1) (Exit Routines)

Chapter 19. IMS Connect two-phase commit support

IMS Connect supports two phase-commit for both IMS TM transactions and IMS DB database requests.

The IMS TM transactions and IMS DB database requests that are participants in two-phase-commit transactions are coordinated by z/OS Resource Recovery Services or an external coordinator (for example, IBM WebSphere Application Server).

When accessing IMS TM through IMS Connect, the external coordinator must use the IMS TM Resource Adapter as the resource adapter. Together, the IMS TM Resource Adapter and IMS Connect handle the data flow for two-phase-commit processing.

When accessing IMS DB through IMS Connect, the external coordinator can use any of the following resource adapters and APIs:

- IMS Universal Database resource adapter
- IMS Universal JDBC driver
- IMS Universal DL/I driver
- The Distributed Relational Database Architecture (DRDA)

To handle the data flow for two-phase-commit processing for IMS DB support, IMS Connect works with the CSL Open Database Manager (ODBM) component, together with the IMS Universal drivers.

This topic provides an overview of two-phase commit and some key scenarios that IMS Connect supports.

Overview of two-phase commit protocol

Two-phase commit protocol is comprised of a set of actions that ensure a transaction involving multiple databases does not produce unsynchronized updates.

Two-phase commit provides a way for a series of database interactions on multiple different data sources to be grouped together and completed or rolled back as a single transaction. Two-phase commit transactions that represent a series of database interactions on multiple data sources are referred to as *global transactions*.

At the beginning of a global transaction, a global transaction ID (XID) is generated and used by an external transaction manager to drive the two-phase commit processing across all of the resource managers involved. Each database interaction within the scope of the global transaction is executed upon its associated resource manager. The results of the interactions are then sent back to the application for processing. When the database interactions within the scope of the global transaction are finished, a prepare call is sent to each resource manager that was accessed by the global transaction. The prepare call provides each resource manager a chance to determine and report on its ability to commit the work it has done as a part of the global transaction. Upon receiving verification that each resource manager can commit its work, the transaction manager sends a commit call to each resource manager.

At any point in time prior to sending the commit call, the two-phase commit transaction can be rolled back. If the transaction is rolled back, a rollback call is sent to each resource manager involved in the transaction and the temporary changes are removed or discarded.

If any database failures occur during the commit phase, the external transaction manager tries to reestablish a connection with the failed resource manager and resumes its call for the resource manager to commit.

If the transaction manager fails during the commit phase, the transaction manager performs recovery processing upon restart and attempts to reestablish a connection with all of the resource managers involved. When the transaction managers reestablishes the connections, it resumes its call for the resource managers to commit.

© Copyright IBM Corp. 1974, 2020

Distributed two-phase commit support

Distributed two-phase commit protocol uses TCP/IP to communicate transactions between various platforms (for example, Windows, AIX®, Solaris, Linux®).

A distributed TCP/IP transaction normally involves the following components:

- An application component
- An application server, such as WebSphere Application Server
- A resource adapter, such as one of the IMS Universal Database resource adapters, or the IMS TM Resource Adapter.
- A resource manager
- A transaction manager, such as IMS TM
- An enterprise information system (EIS), such as IMS DB

In distributed two-phase commit protocol, a client that is deployed on an application server issues a transaction. The application server acts as an external transaction manager (external coordinator) to manage transactions across one or more resource managers. To access the resource manager of an enterprise information system, the external coordinator must use a resource adapter. The IMS Connect client accesses the resource manager (IMS) through IMS Connect using TCP/IP.

IMS supports the X/Open XA protocol through z/OS Resource Recovery Services (RRS), which IMS uses to participate in two-phase-commit processing on z/OS. As a result, IMS Connect communicates with RRS and passes to RRS the transaction context from the IMS Connect client. In turn, RRS as the syncpoint coordinator coordinates the changes so that all or no updates are made to IMS. In RRS, the set of changes that are made or not made within the transaction scope is called a unit of recovery (UR).

IMS Connect plays dual roles in two-phase-commit processing. IMS Connect, acts as an extension to RRS (the syncpoint manager) and is considered the server distributed syncpoint resource manager (SDSRM). As the SDSRM, IMS Connect allows RRS to communicate with other syncpoint managers as needed to ensure coordination of the distributed resources the application accesses. IMS Connect also is the communication resource manager (CRM). As the CRM, IMS Connect controls access to distributed resources by allowing an application component to communicate with other application components and resource managers that may be on different systems. Also, as the CRM, IMS Connect assists in processing a syncpoint event and communicates the events to distributed syncpoint managers.

IMS distributed transaction processing can be broken down into two types of transactions: global transactions that use two-phase commit protocol and transactions that use the one-phase commit optimization. The one-phase commit optimization is used when only one resource manager is involved in the transaction.

Support for the IMS Universal drivers

IMS Connect supports two phase-commit and one-phase commit for IMS DB database requests that are received from IMS Universal drivers.

Global (XA) transactions with the IMS Universal drivers

IMS support for global, two-phase commit transactions for the IMS Universal drivers uses z/OS Resource Recovery Services (RRS) multisystem cascaded transactions to coordinate database access requests between IMS Connect, the CSL Open Database Manager (ODBM), and IMS within an IMSplex.

The IMS Universal drivers provide the LocalTransaction and XAResources interfaces for communication with IMS Connect. IMS Connect maintains the XID and associates it with the work context token and the parent unit of recovery (UR), which represents the transaction as submitted by the IMS Universal drivers.

When IMS Connect receives a database access request from any of the IMS Universal drivers that requires two-phase commit, IMS Connect calls RRS and becomes the *coordinator* of the multisystem cascaded transaction. IMS Connect creates a parent unit of recovery (UR) and passes a parent UR token with the database access request to the CSL Open Database Manager (ODBM).

ODBM becomes a *subordinate* to the multisystem cascaded transaction and uses the parent-UR token received from IMS Connect to create a cascaded child UR.

When the client application requests prepare for commit, the IMS Universal drivers send a prepare signal to IMS Connect. IMS Connect then issues a prepare request to RRS, and waits for ODBM to complete phase one of the two-phase commit process. If the IMS resource manager is prepared to commit, ODBM returns the prepare to commit confirmation to RRS and RRS sends the results to IMS Connect. IMS Connect then sends the result of the prepare command back through the IMS Universal drivers to the transaction manager. At this point, the UR maintained by IMS Connect and ODBM is in an indoubt state until a commit or rollback command is sent in from the external transaction manager.

If all the resource managers associated with the global transaction can commit, the transaction manager hardens the commit decision and drives the IMS Universal drivers to commit the change. The IMS Universal drivers send a commit signal to IMS Connect and IMS Connect tells RRS that the overall decision is to commit all resources. RRS tells IMS to commit the changes. After IMS commits the changes, RRS then returns to IMS Connect with the information that the local resources have been committed. IMS Connect tells RRS to delete its log records.

The following figure illustrates the first 17 steps of the flow of a distributed two-phase commit global transaction that is submitted to IMS Connect from the IMS Universal drivers. The transaction involves one IMS system. RRS needs to be active on each z/OS image; however, IMS Connect does not need to be on the same z/OS image as ODBM and IMS.

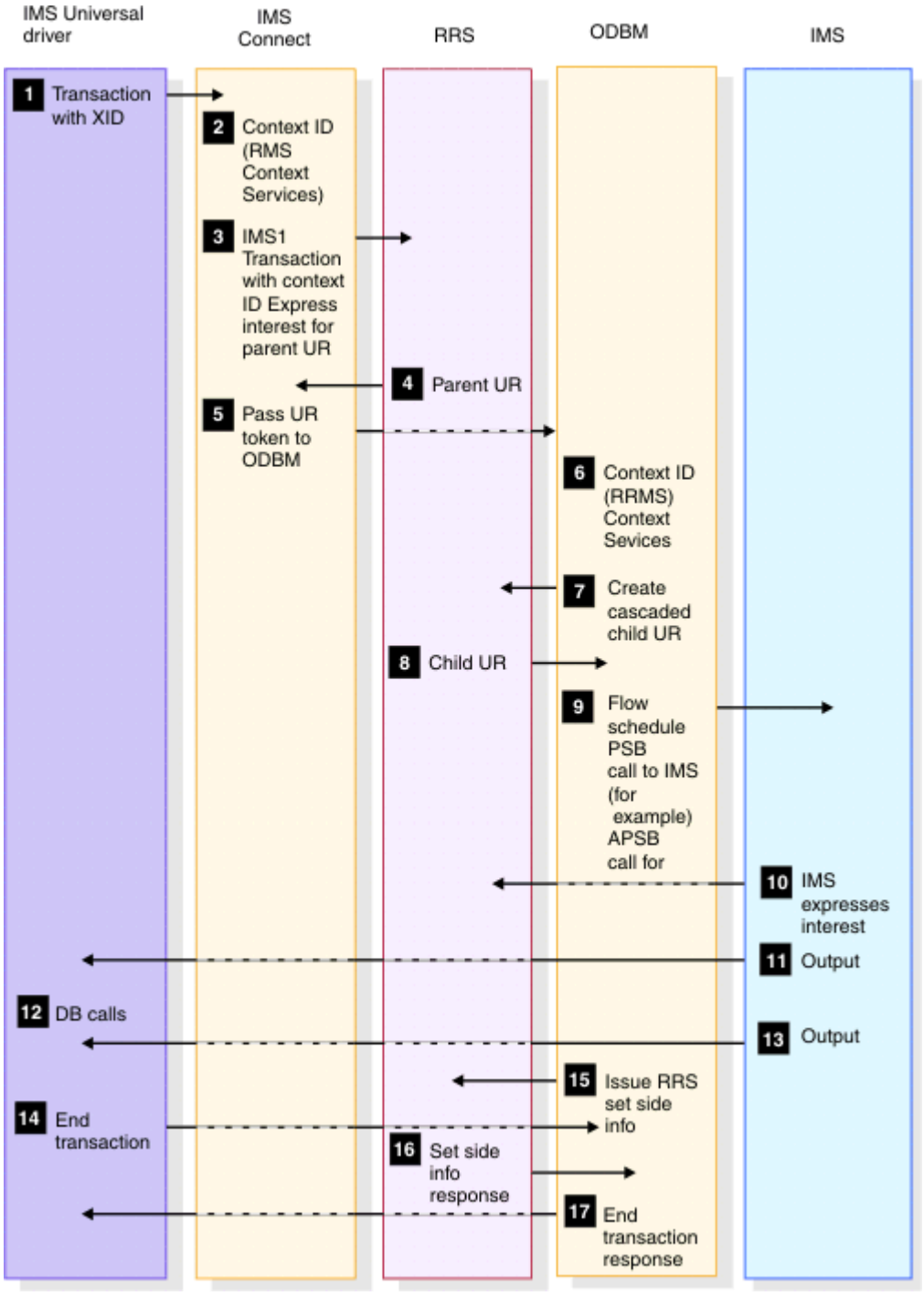


Figure 35. Distributed two-phase commit global transaction client flow for IMS Universal drivers (1 of 2)

The following figure illustrates the remaining steps of the flow of the distributed two-phase commit global transaction shown in the preceding figure.

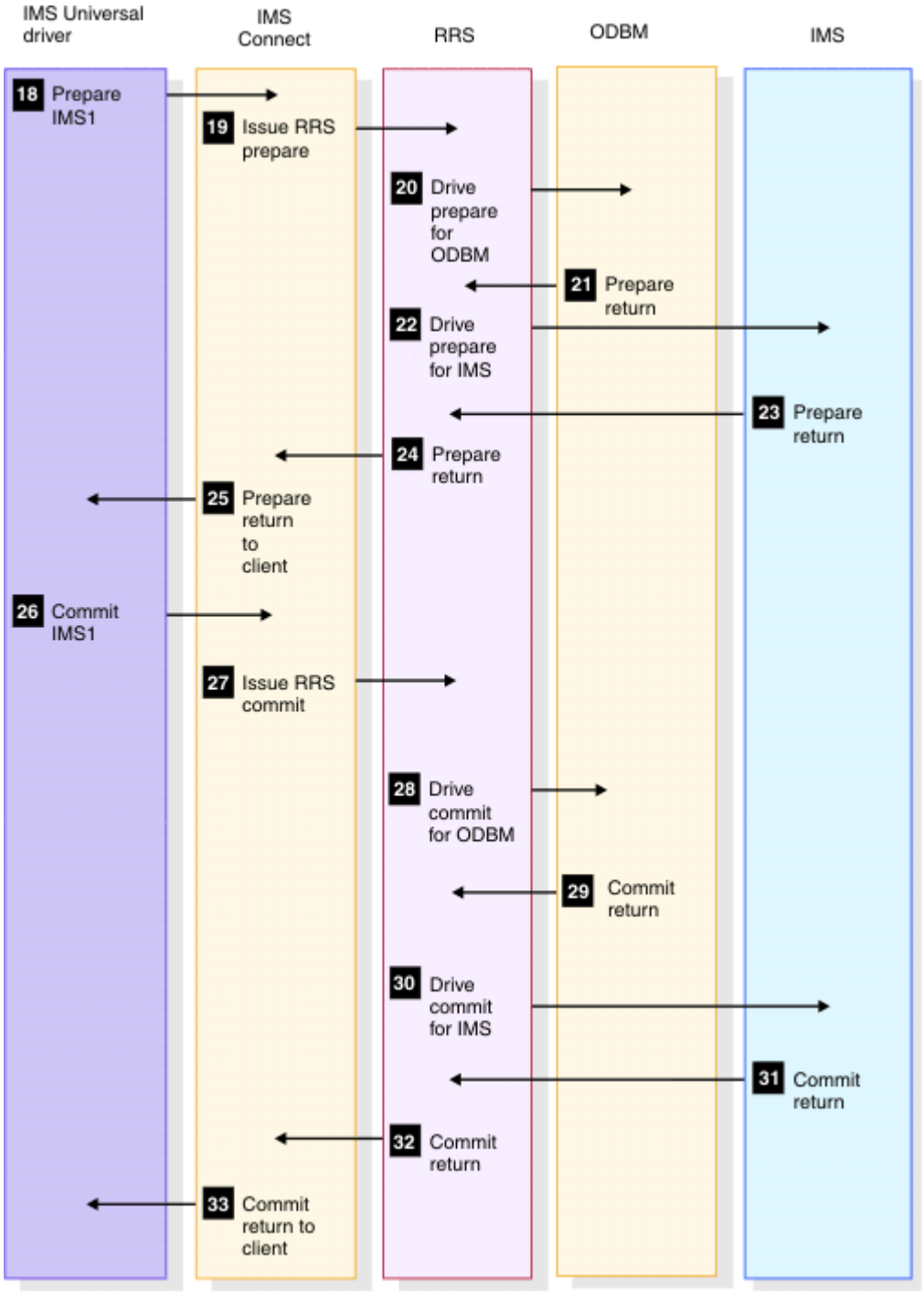


Figure 36. Distributed two-phase commit global transaction client flow for IMS Universal drivers (2 of 2)

One-phase commit global transactions with the IMS Universal drivers

If the external transaction manager detects that only one resource manager is enlisted in a given transaction, the transaction manager can perform one-phase-commit optimization.

In the one-phase-commit optimization, the first phase of the two-phase commit operation, the prepare phase, is omitted and the transaction manager sends only the *commit request* to the resource manager to commit the changes without first sending *prepare to commit*.

z/OS Resource Recovery Services (RRS) transactions that use one-phase commit are also referred to as an RRS *local transaction*. Local transactions do not contain an XID.

When IMS Connect receives from the IMS Universal drivers a database access request that is an RRS local transaction, IMS Connect does not call RRS and passes the local transaction to ODBM directly. ODBM in turn calls RRS and coordinates the commit or rollback of the database request.

The following figure illustrates the flow for a distributed one-phase commit global transaction. RRS needs to be active on each z/OS image; however, IMS Connect does not need to be on the same z/OS image as ODBM and IMS.

The following figure illustrates the first 17 steps of the flow of a distributed one-phase commit global transaction that is submitted to IMS Connect from the IMS Universal drivers. The transaction involves one IMS system. RRS needs to be active on all z/OS images; however, IMS Connect does not need to be on the same z/OS image as ODBM and IMS.

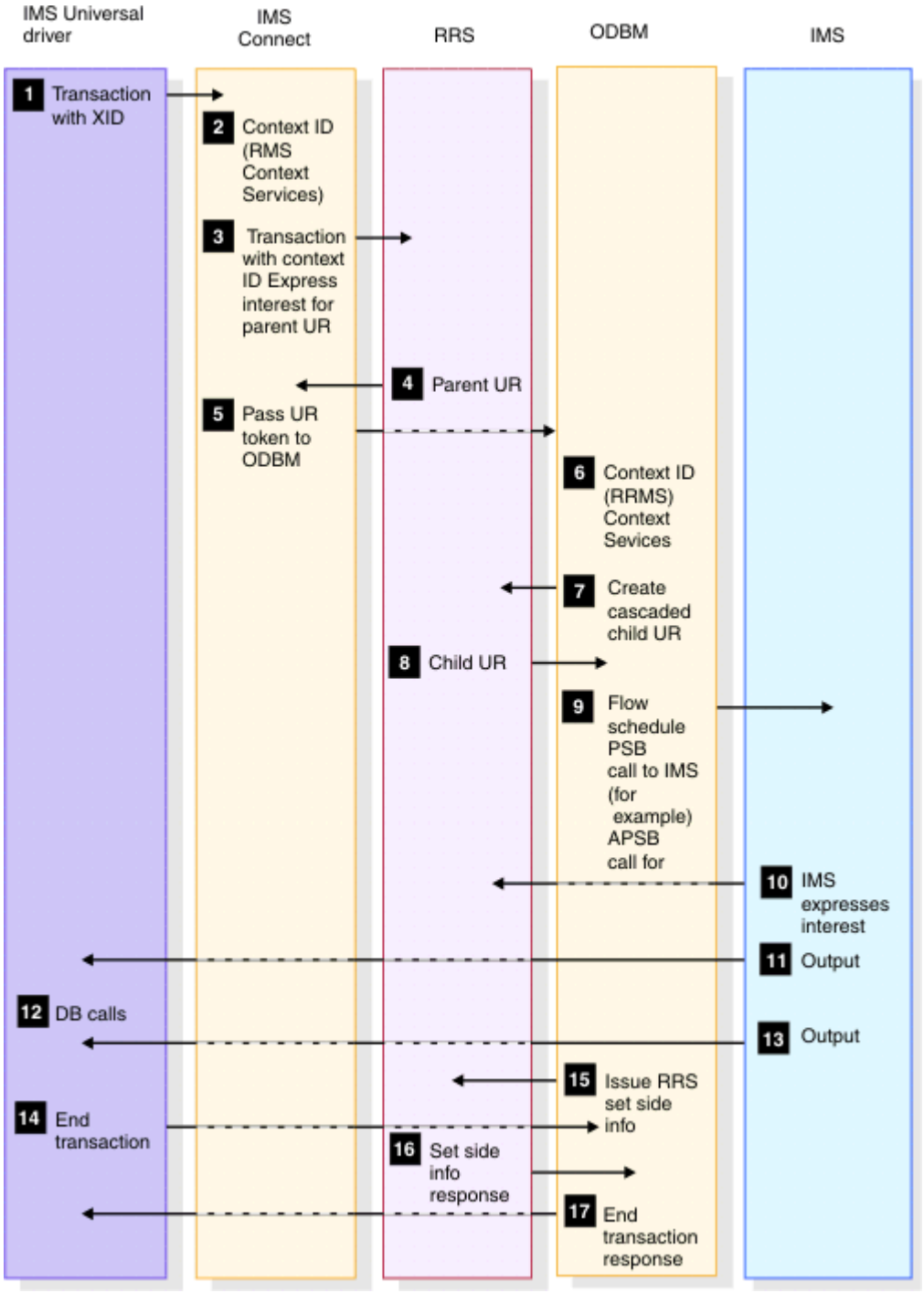


Figure 37. Distributed one-phase commit global transaction client flow for IMS Universal drivers (1 of 2)

The following figure illustrates the remaining steps of the flow of the distributed one-phase commit global transaction shown in the preceding figure.

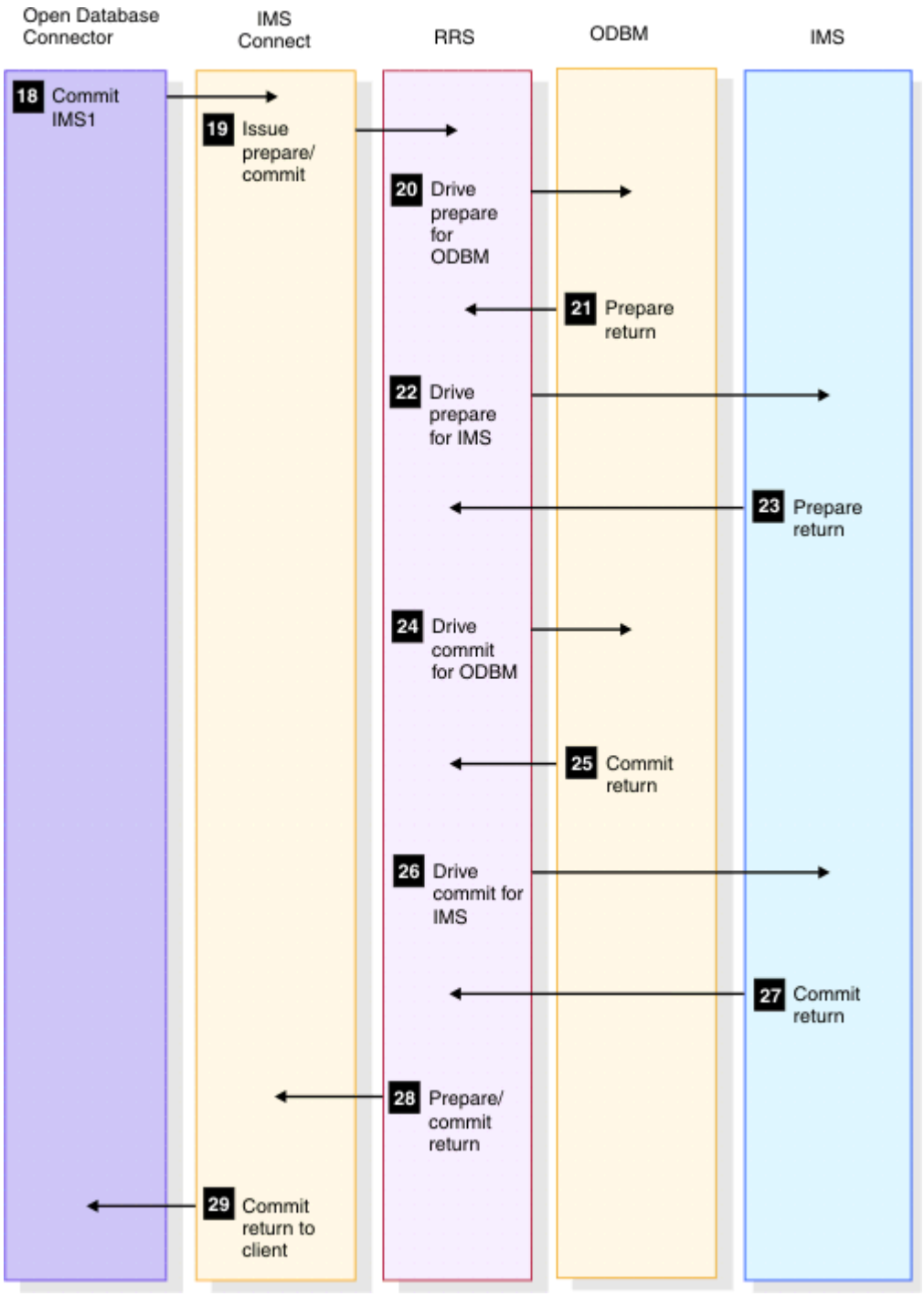


Figure 38. Distributed one-phase commit global transaction client flow for IMS Universal drivers (2 of 2)

Support for the IMS TM Resource Adapter

IMS Connect supports two phase-commit and one-phase commit for IMS TM transactions that are received from the IMS TM Resource Adapter.

Global (XA) transactions with IMS TM Resource Adapter

A global (XA) transaction is controlled and coordinated by an external transaction manager (external coordinator) to a resource manager.

The transaction normally requires coordination across multiple resource managers that may reside on different platforms. The transaction must be sent to IMS Connect as send-then-commit (commit-mode 1) with a sync level of syncpoint.

To access an enterprise information system, the external coordinator sends an XID, which is defined by the X/Open XA standard, to a resource adapter. In addition to the length and FormatID fields, an XID has two other parts: the global transaction identifier (GTRID) and the branch qualifier (BQUAL). Because IMS does not support X/Open XA protocol, the IMS TM Resource Adapter uses the LocalTransaction and XAResources interfaces to participate in transactions coordinated by the external coordinator to communicate with IMS Connect. IMS Connect maintains the XID and associates it with a work context token and an IMS name. IMS Connect then passes the context token to z/OS Resource Recovery Services (RRS).

IMS Connect sends the transaction output back to the IMS TM Resource Adapter which returns the output data to the client. Upon sending the output message to the IMS TM Resource Adapter successfully, IMS Connect sends an ACK to IMS to acknowledge the message. After making requests to IMS, the application component indicates to the IMS TM Resource Adapter that it is ready to commit the changes. At this point the IMS TM Resource Adapter sends a prepare signal to IMS Connect. IMS Connect, in turn, tells RRS to initiate the prepare phase. If the IMS resource manager is prepared to commit, RRS collects the prepare to commit confirmation from the resource manager and sends the results to IMS Connect. IMS Connect will then send a *request to commit* signal to the IMS TM Resource Adapter to request committing the changes.

When the IMS TM Resource Adapter receives the request to commit signal, it tells the external coordinator that the resources on the IMS system can be committed. The transaction manager determines the overall results. If all the resource managers can commit, the transaction manager hardens the commit decision and will drive the IMS TM Resource Adapter to commit the change. The IMS TM Resource Adapter sends a commit signal to IMS Connect and IMS Connect tells RRS that the overall decision is to commit all resources. RRS tells IMS to commit the changes. After IMS commits the changes, RRS then returns to IMS Connect with the information that the local resources have been committed. IMS Connect tells RRS to delete its log records.

If support for cascading global transactions is enabled by the specification of CASCADE=Y in either the IMS Connect system configuration or IMS Connect data store definitions, IMS Connect and IMS can process a global transaction when IMS Connect and IMS are each running on different z/OS images (LPARs). When support for cascading global transactions is disabled, IMS Connect, RRS, and IMS must all be on the same LPAR.

Requirement: All message flows of a global transaction, which can include multiple separate transactions submitted to IMS from the same global transaction, must be processed by the same IMS and IMS Connect pair. Because of this requirement, you might not be able to send global transactions to IMS if you use software that distributes workload across multiple instances of IMS Connect, such as the z/OS Sysplex Distributor. Such workload distribution software is not likely to guarantee that all flows from the same global transaction will be routed to the same IMS and IMS Connect pair.

The following series of figures illustrate the flow of distributed two-phase commit global transactions between IMS TM Resource Adapter and IMS in different configuration scenarios.

The following figure shows the flow of a single two-phase commit transaction between IMS TM Resource Adapter and a single IMS system. In the figure, IMS Connect and IMS are running on the same LPAR.

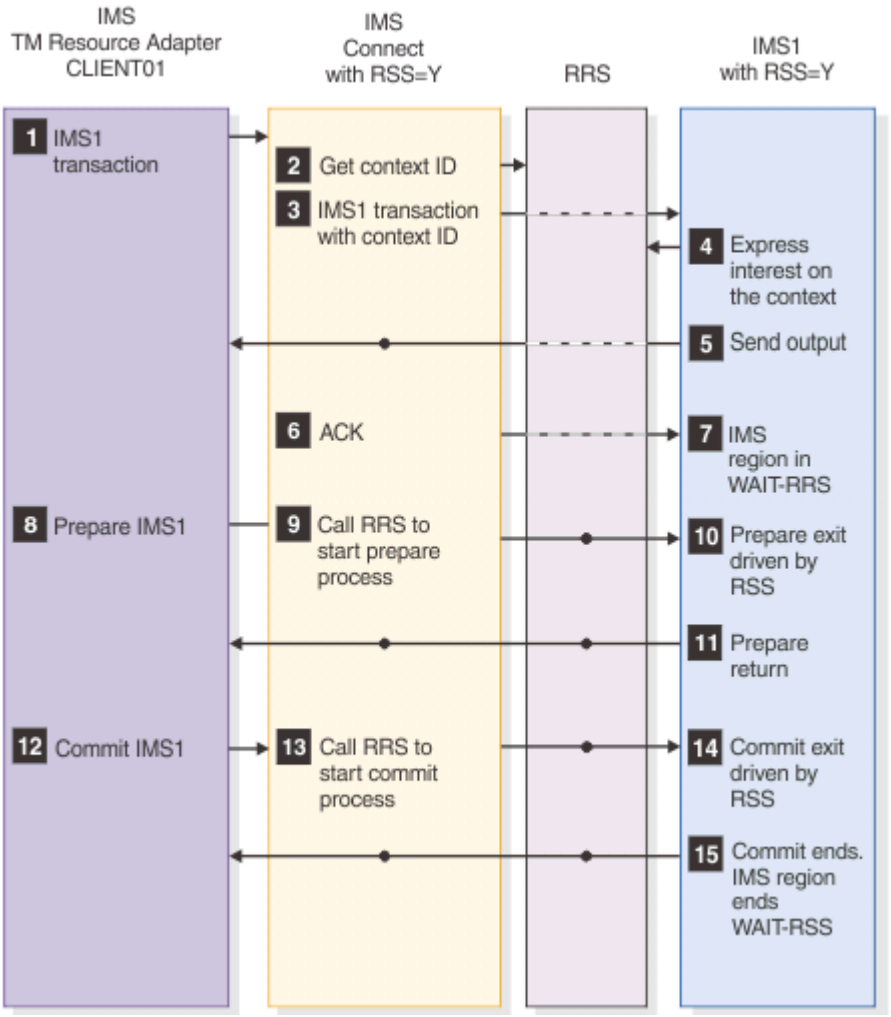


Figure 39. Flow of a single distributed two-phase commit global transaction

The following figure shows the flow of a single two-phase commit transaction between IMS TM Resource Adapter and a single IMS system. In the figure, IMS Connect and IMS are running on different LPARs.

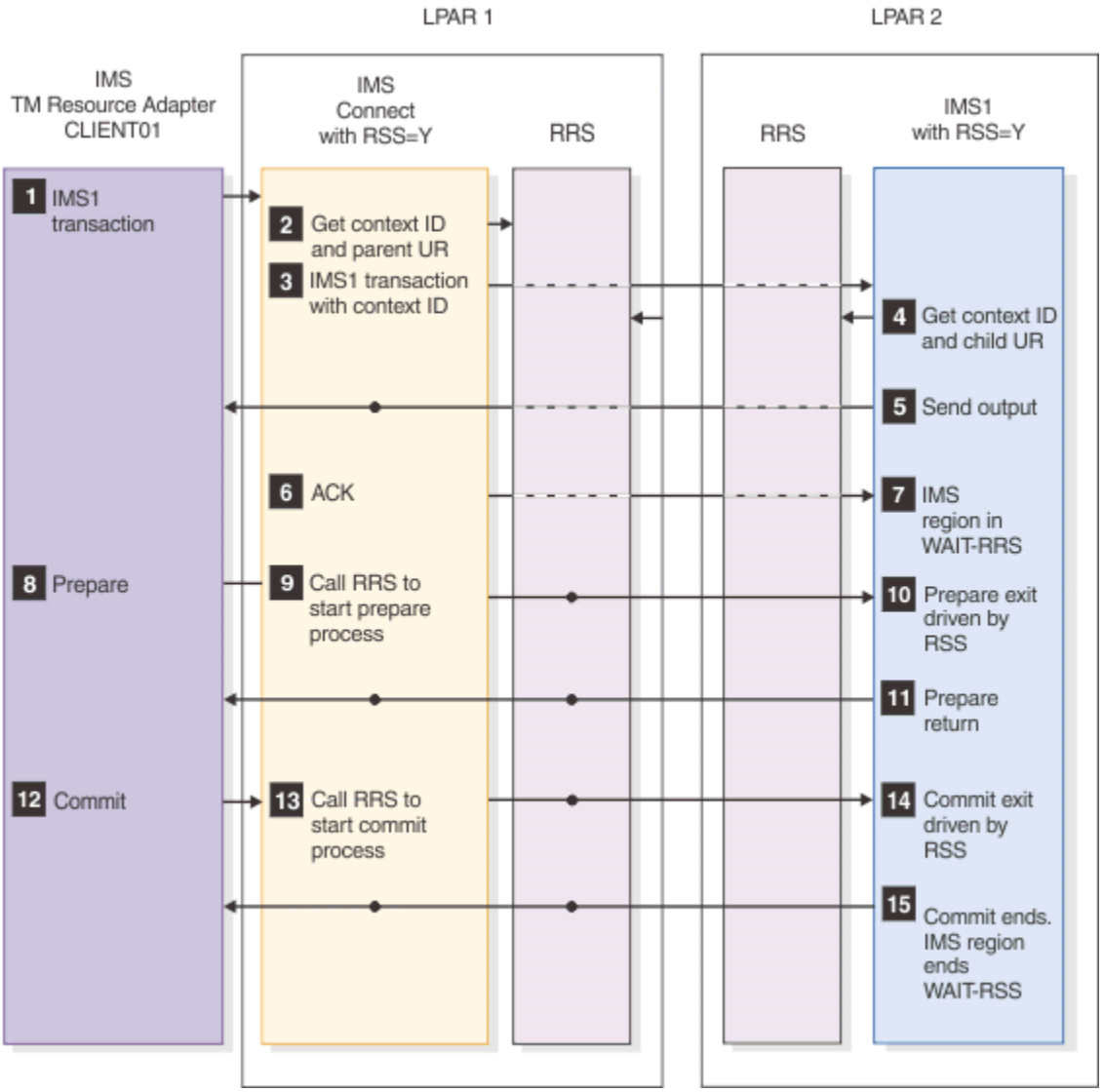


Figure 40. Flow of a single, cascaded distributed two-phase commit global transaction

The following figure illustrates the flow of a distributed two-phase commit global transaction. The transaction involves two IMS systems.

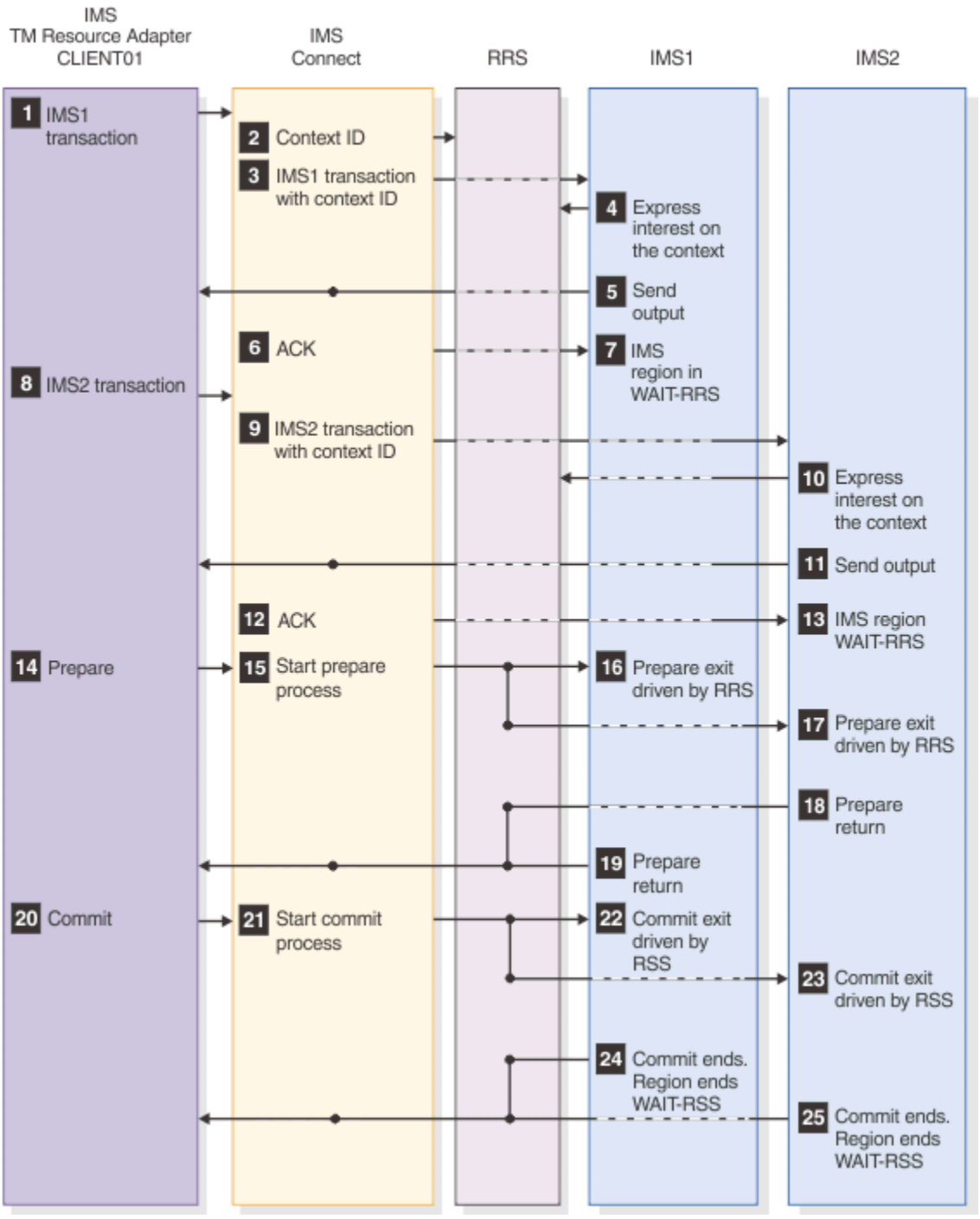


Figure 41. Distributed two-phase commit global transaction client flow

Cascading global transactions from IMS TM Resource Adapter to IMS systems on different z/OS images

IMS Connect can cascade global (XA) transactions that are submitted on a TCP/IP connection by the IMS TM Resource Adapter to IMS TM systems that are running on different z/OS images (LPARs).

By default, support for cascading global transactions from the IMS TM resource adapter to IMS systems on different LPARs is disabled. Support for cascading transactions is enabled by specifying the CASCADE parameter in either the definition of the data store connection or the definition of the IMS Connect system configuration.

Performance for global transactions is best when IMS and IMS Connect are running on the same LPAR, so unless performance is not a primary concern, cascade global transactions across LPARs only temporarily as failover protection for when an IMS system that is on the same LPAR as IMS Connect is unavailable.

You can enable IMS Connect support for cascading global transactions to IMS TM data stores that are located on different LPARs in any of the following ways:

1. Set the IMS Connect system default support by either:
   - Issuing the IMS type-2 command UPDATE IMSCON TYPE(CONFIG) SET(CASCADE(ON))
   - Specifying CASCADE=Y in the HWS configuration statement in the HWSCFGxx member of the IMS PROCLIB data set
2. Set the support option for an individual data store connection in any of the following ways:
   - Issuing the IMS type-2 command CREATE IMSCON TYPE(DATASTORE) SET(CASCADE(ON))
   - Issuing the IMS type-2 command UPDATE IMSCON TYPE(DATASTORE) SET(CASCADE(ON))
   - Specifying CASCADE=Y in the DATASTORE configuration statement in the HWSCFGxx member of the IMS PROCLIB data set
3. If you updated an existing data store connection, restart the data store connection after the CASCADE option is set.

One-phase commit global transactions with IMS TM Resource Adapter

If only one resource manager is registered in a transaction that is making changes to shared resources, the transaction manager can perform one-phase-commit optimization. An external coordinator is not required.

The transaction manager can send the phase two *commit request* directly to the resource manager to commit the changes. IMS Connect does not have to go through phase one, *prepare to commit* of the two-phase commit protocol and can go directly to phase two, *commit request*.

If support for cascading global transactions is enabled by the specification of CASCADE=Y in either the IMS Connect system configuration or IMS Connect data store definitions, IMS Connect and IMS can process a global transaction when IMS Connect and IMS are each running on different z/OS images. When support for cascading global transactions is disabled, IMS Connect, RRS, and IMS must all be on the samez/OS image.

The following figure illustrates the flow for a distributed one-phase commit global transaction.

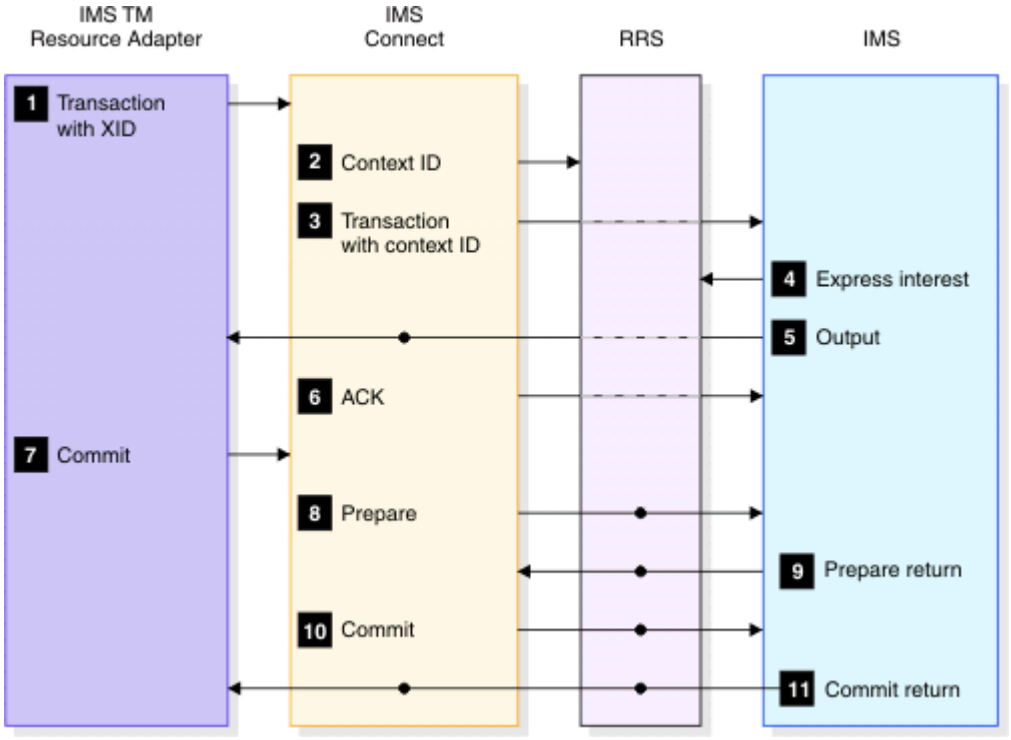


Figure 42. Distributed one-phase commit optimization client flow

Chapter 20. Unicode considerations for IMS Connect

IMS Connect support for Unicode allows Unicode data to be sent to and from an IMS Connect client application. In certain circumstances, IMS Connect translates the Unicode data.

The IMS Connect support for Unicode requires that the client application and the IMS host application both support:

- Unicode data
- The same Unicode encoding schema (either UTF8, UTF16, or UCS-2) as the structure and content of the message being sent and received

IMS Connect supports ASCII and EBCDIC data streams both to and from the client application. If the client application sends ASCII data to IMS Connect, the ASCII is translated to EBCDIC. The subsequent output from IMS Connect to the client application is translated back to ASCII from EBCDIC. If the client application sends EBCDIC data to IMS Connect, no translation is required. With Unicode support, an IMS client application can also send and receive Unicode data to and from IMS Connect, specifically UTF8, UTF16, or UCS-2 data streams.

IMS Connect supports language groups 1, 2, and 3.

The client application uses the IMS Request Message (IRM) to:

- Tell IMS Connect if the data it is sending is Unicode and if the IMS transaction code is being sent as Unicode. IMS Connect transforms the transaction code if it is being sent as Unicode and then sends the transformed code and Unicode data to IMS.
- Tell IMS Connect the Unicode encoding schema that is being used (UTF8, UTF16, or UCS-2).

The transaction code can be sent as Unicode, ASCII, or EBCDIC; however, it must be a valid IMS transaction code of up to 8 bytes that occupies an 8-byte field. In the field, the code must be left justified and, if it is shorter than 8 bytes, padded with blanks. If a blank follows the 8-byte transaction code field, it is considered to be part of the Unicode data.

Message translation

All IMS error messages (for example, DFS555) are sent as either ASCII or EBCDIC. The client application uses the IRM_ID field of the IMS request message (IRM) header to tell IMS Connect which type to send. IMS Connect does not transform messages to Unicode.

For example, if IRM_ID is EBCDIC, the IMS error message (DFSnnnn) is sent as EBCDIC; if IRM_ID is ASCII, the IMS error message (DFSnnnn) is translated from EBCDIC to ASCII.

IRM_ID also identifies the code type of the OTMA header.

An IMS client application can send the IMS transaction code as ASCII, EBCDIC, or Unicode. When the IMS client application sends the transaction code as Unicode, the IMS Connect user message exit (HWSSMPL0 and HWSSMPL1) translates the transaction code from Unicode TO EBCDIC. When the client application sends the transaction code as ASCII and the remaining data as Unicode, only the transaction code is translated to EBCDIC. A valid 8-byte IMS transaction code can be constructed from the following characters and must begin with an alphabetic character:

- A through Z (uppercase only)
- 0 through 9
- Special characters #, $, @

An IMS host application that supports Unicode must define an 8-byte field in the input message definition to contain the transaction code. If you pad the 8-byte field with a blank, it is sent as an EBCDIC blank.

If the client application sends Unicode data, the output message is not transformed and is treated as Unicode. For RESUME TPIPE calls, the client application must specify in the IRM if the output should

© Copyright IBM Corp. 1974, 2020

be treated as Unicode or not. During message switching, the IMS host application must ensure that the output message is formatted correctly (using a specific Unicode schema or EBCDIC) for its destination.

Input message format sent by the client

The following table contrasts the message structure for input messages sent by the client and defines the valid ASCII, EBCDIC, and UNICODE formats.

Table 69. Input message structure - message sent by client

EBCDIC IRM	ASCII IRM	If OTMA headers are passed by client	Transaction code	Data
Y	N/A	EBCDIC	EBCDIC	UNICODE
Y	N/A	EBCDIC	UNICODE	UNICODE
N/A	Y	ASCII	ASCII	UNICODE
N/A	Y	ASCII	UNICODE	UNICODE

Output message format received by the client

The following table defines the valid output message elements when the client sends UNICODE data.

Table 70. Output message structure - message received by client

If input message was EBCDIC IRM	If input message was ASCII IRM	RMM	RSM	Output CSM	Output data
Y	N/A	EBCDIC	EBCDIC	EBCDIC	UNICODE
N/A	Y	ASCII	ASCII	ASCII	UNICODE

Related reference

“Message structures and IMS Connect user message exit routines” on page 231
IMS Connect allows up to 254 user exits to be defined in the configuration file. There are two input message structures supported by IMS Connect and two message structures supported on return from a user exit.

Chapter 21. TCP/IP settings for IMS Connect

You can choose different TCP/IP values to maximize your environment settings for IMS Connect.

The z/OS Communications Server configuration settings for TCP/IP are located in the z/OS PROFILE.TCPIP data set.

The following TCP/IP parameter values affect IMS Connect:

TCPNODELAY=ENABLE

- Data is transmitted by TCP/IP per client SEND.
- TCP/IP waits one millisecond per transmission.
- Multiple client TCP/IP SENDS can result in multiple TCP/IP transmissions.

TCPNODELAY=DISABLE

- Data is collected by TCP/IP from client TCP/IP SENDS, before transmission.
- TCP/IP waits until the buffer is full before transmission.
- Multiple client SENDS results in 1 to *n* TCP/IP transmissions to IMS Connect.

SO_LINGER=Y, VALUE=0

- Immediate return to client code.
- A client request to close the socket can bypass data sent with a previous client TCP/IP SEND request, but may result in the loss of the client SEND data.

SO_LINGER=N

- Immediate return to client code.
- A client request to close the socket can bypass data sent with a previous client TCP/IP SEND request, but may result in the loss of the client SEND data.

SO_LINGER=Y, VALUE=10

- Return to client code when an ACK is received from the host, or wait for 10 seconds before sending close.
- Socket close will not bypass data sent.

DELAYACK

DELAYACK is used to minimize non-data transmissions from the host. If DELAYACK is used, the z/OS TCP/IP waits 200 milliseconds before sending an ACK to the remote server TCP/IP. However, if the ACK is appended to the data being sent from IMS Connect, there is no delay.

If your client application performs a single SEND followed by a READ, DELAYACK is recommended.

DELAYACK can be set on the TCP/IP "Port Statement" or on the "Gateway Statement."

NODELAYACKS

NODELAYACKS is used to allow non-data transmissions from the host to flow without data. If NODELAYACKS is used, the z/OS TCP/IP immediately sends an ACK to the remote server TCP/IP. The ACK is not appended to the data being sent from IMS Connect.

If the client code sends one SEND followed by a READ to the host with a NODELAYACKS setting, an ACK is sent separately.

If the client code sends two or more SENDs followed by a READ to the host, the host TCP/IP will send an ACK immediately to the data received. This will allow the next SEND of data from the client to flow.

NODELAYACKS is recommended if your client application sends more than one SEND followed by a READ.

NODELAYACKS can be set on the TCP/IP "Port Statement" or on the "Gateway Statement."

© Copyright IBM Corp. 1974, 2020

SOMAXCONN

The SOMAXCONN statement can be used in conjunction with the TCPIPQ parameter of the TCPIP statement in the IMS Connect HWSCFG*xx* member of the IMS.PROCLIB data set. The SOMAXCONN statement in PROFILE.TCPIP controls the TCP/IP queue depth for listening sockets at the LPAR level. You can use the TCPIPQ parameter to override the value of SOMAXCONN for an instance of IMS Connect. The value of TCPIPQ is used only if it is lower than the value of SOMAXCONN for the host LPAR.

Recommendation: Because the default value of SOMAXCONN (10) is lower than the minimum value of TCPIPQ (50), use the TCPIPQ parameter only when the value for SOMAXCONN is modified to be greater than 50.

Part 6. IMS VTAM network administration

This describes information that will help you to administer your IMS network.

© Copyright IBM Corp. 1974, 2020

Chapter 22. Introduction to the IMS Transaction Manager network

The following topics introduce the major IMS Transaction Manager (TM) network components, and describes the way communications are established, messages are transmitted, and communications are terminated between IMS and a logical unit.

IMS TM network overview

IMS is a general-purpose database and transaction manager system that provides the necessary support for an advanced telecommunications network. Virtual Telecommunications Access Method (VTAM) controls the physical transmission of data in the network, directing data to IMS from various logical units and from IMS to the appropriate logical units.

An IMS telecommunications network must include the following components:

- IMS
- VTAM (Although IMS uses the network facilities of VTAM, it can also control non-VTAM devices such as Basic Sequential Access Method (BSAM). VTAM is the preferred access method for IMS.)
- Communications hardware (such as control units)
- Terminals

The network can optionally include any of the following components:

- IMS Transaction Manager (IMS TM) services, such as:
  - Extended Terminal Option (ETO)
  - Fast Path
  - Message Format Service (MFS)
  - Intersystem Communication (ISC)
  - Multiple Systems Coupling (MSC)
  - Advanced Program-to-Program Communication (APPC)
- Common Queue Server (CQS) and a coupling facility with any of the following structures:
  - Shared-queues structures
  - Shared-data structures
  - Resource structure
- VTAM generic resource groups
- Open Transaction Manager Access (OTMA)
- Common Service Layer (CSL) including:
  - Operations Manager (OM)
  - Resource Manager (RM)
  - Structured Call Interface (SCI)

In addition, an IMS telecommunications network can operate within one of the following frameworks:

- IBM Systems Network Architecture (SNA), which brings together multiple products in a unified design. SNA formally defines the functional responsibilities of the network components.
- A client-server environment, using Advanced Program to Program Communications (APPC) or OTMA.

Definitions:

© Copyright IBM Corp. 1974, 2020

- A *logical unit* is an addressable resource that can be an application program, a terminal, or a subsystem such as Customer Information Control System (CICS). A logical unit can also be a component of a general-purpose terminal system that consists of a programmable controller and its attached operator terminals, printers, and auxiliary control units.
- The word *terminal* is used throughout this manual to describe devices and also applies to controllers and remote subsystems. The operator terminals can be keyboard printers, display stations with keyboards, communication terminals, or a mixture of these devices.

A network consisting of IMS and programmable logical units enables users to distribute functions throughout network components. This distribution of function reduces processing requirements that are placed on the central processor (also referred to as the *host*), and it can reduce the impact on the rest of the network when one component encounters a problem.

The following figure illustrates the components of a complete communications network system. The arrows in the figure indicate communications that occur between components. The figure shows the following components:

- IMS and its application programs
- Virtual Telecommunications Access Method (VTAM)
- Tivoli NetView for z/OS
- z/OS operating system (including APPC/MVS if APPC/IMS is used)
- IBM 37x5 Communications Controller and Network Control Program (NCP)
- Terminal

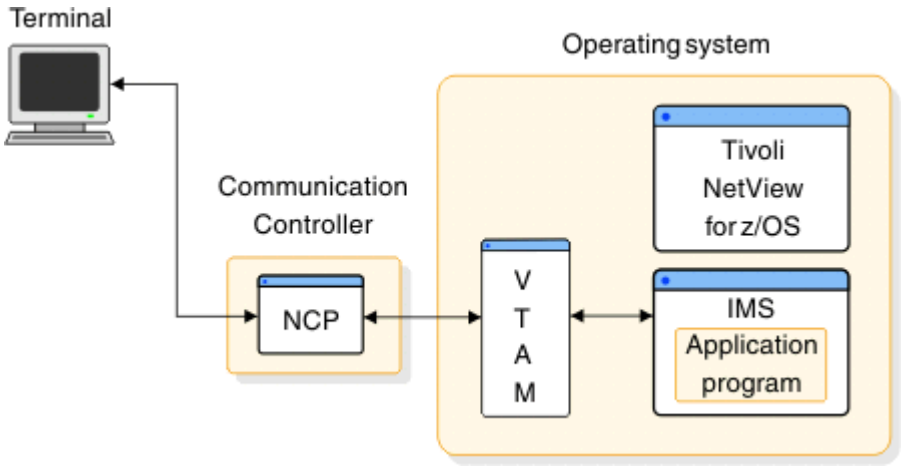


Figure 43. Components of a network

The following list summarizes how each of the components in the preceding figure participates in the network:

IMS

- Checks transaction security
- Schedules the proper application program
- Directs output to the proper terminal
- Provides checkpoint and recovery capabilities

Application program

- Reads data from terminal
- Writes data to processor
- Reads data from processor

- Writes data to terminal

VTAM

- Connects and disconnects terminal from the network
- Sends data from terminal to IMS
- Permits both monitoring and modifying of the network
- Sends data from IMS to the terminal
- Manages physical network (with Tivoli NetView for z/OS)

Communications Controller

- Adds line control characters
- Transmits data
- Receives data
- Removes line control characters
- Checks for transmission errors
- Controls buffering

Network Control Programs (NCP)

- Sends and receives data from communication lines and adapters
- Checks for and records errors

Planning an IMS network requires an understanding of each component and of its relationship to the others.

The following topics provide additional information:

Programmable logical unit (LU)

Definitions:

- A *programmable logical unit (LU)* is an input/output device that is in session with IMS. Application programs in remote logical units can be designed to control more than one terminal.
- When a logical unit informs VTAM that it wants to communicate with IMS, VTAM notifies IMS using the VTAM Logon exit routine. IMS then accepts the request, and VTAM logically connects the logical unit to IMS. This logical connection is called a *session*.

Some of the functions performed by the remote application program include:

- Reading from and writing to associated terminals
- Editing and verifying the data that is received from a terminal
- Reading from and writing to disk storage within the remote LU
- Reading from and writing to the host in which IMS is running
- Editing and verifying data that is received from the host in which IMS is running
- Communicating with other network logical units
- Formatting display and printer devices
- Operating offline when the host, VTAM, IMS, or NCP is unavailable

Depending on the system type, each LU can consist of one or more terminals. An application program that controls an LU consisting of more than one terminal or component must be able to direct output to a

specific device. Therefore the application program must be capable of some form of data interrogation in order to make the proper device selection. IMS assists the application program in this process by:

- Allowing LU components to be defined and addressed individually
- Providing, in the header of each output message, the identification of the component to which the message is directed

Communications Controller and Network Control Program (NCP)

VTAM uses the facilities of NCP, which runs in the 37x5 Communications Controllers. VTAM uses NCP to:

- Control lines and devices that are attached to the controllers
- Transmit data between the logical unit and the host CPC
- Perform error recovery
- Collect statistics about the network

Virtual Telecommunications Access Method (VTAM)

VTAM controls the allocation of network resources, and enables these resources to be shared among VTAM users. To VTAM, IMS is a single VTAM user; VTAM is unaware of the IMS application programs.

The IMS application programs use the IMS CALL interface to request IMS services; IMS uses VTAM macros in order to activate VTAM facilities.

If APPC/IMS is active, VTAM regards it as a separate user.

Using VTAM USERVAR with IMS

IMS uses VTAM user variables (USERVARs) in XRF complexes to help maintain and manage sessions when an active IMS subsystem fails. You can, in limited cases, also use VTAM USERVARs to point to a VTAM APPLID of an IMS that is not part of an XRF complex; however, IMS only supports this use of VTAM USERVARs if the IMS being pointed to is the session's primary logical unit (PLU).

For example, you might use a VTAM USERVAR in a non-XRF context if you change the APPLID of an IMS system, but want to temporarily allow LUs to continue connecting to the IMS system using the old APPLID.

Note: In an ISC or MSC environment, a VTAM USERVAR might behave unpredictably when pointing to an IMS subsystem that is not part of an XRF complex. This unpredictable behavior is due to the fact that IMS subsystems can be either the PLU or the secondary logical unit (SLU) in sessions between two IMS subsystems in ISC and MSC environments.

Related Reading:

- For more information on VTAM and how it is used, see *z/OS Communications Server: SNA Programming*.
- For more information on VTAM USERVARs, see the *z/OS Communications Server: SNA Network Implementation Guide*

IMS product features

IMS comprises two main product features:

- The Database Manager (IMS DB), which can control your databases
- The Transaction Manager (IMS TM), which can control your data communications and application programs

IMS provides:

- Standard functions required by application programs

- An execution environment for concurrently running application programs that serve many online users
- Control of full-function and Fast Path databases

IMS and the application programs that it controls run under z/OS.

Related concepts
Extended Recovery Facility Overview (System Administration)

IMS Transaction Manager services

This topic describes several specialized optional Transaction Manager services.

APPC/IMS and LU 6.2 devices

APPC/IMS is a part of IMS TM that allows you to use Common Programming Interface Communications (CPI-C) to build CPI application programs. APPC/IMS supports APPC with facilities provided by APPC/MVS.

IMS supports implicit and explicit application program interfaces (APIs) for APPC support. The implicit API for APPC support allows the IMS application program to communicate with APPC devices using the same techniques employed with other remote devices. This support allows an application program, written by a programmer who has no knowledge of APPC LU 6.2 devices, to:

- Be started from an APPC LU 6.2 device
- Receive input messages from originating LU 6.2 devices
- Send output messages back to originating LU 6.2 devices
- Start transaction programs on remote LU 6.2 devices

The same application program can work with all device types (LU 6.2 and non-LU 6.2) without new or changed coding.

The explicit application programming interface (API) for APPC support is the CPI-C interface, and it is available to any IMS application program. The IMS application program can issue calls [9] to APPC/MVS through this interface.

Fast Path

Fast Path is capable of performing transaction and database processing at high rates. When your system requirements include a high transaction volume, using Fast Path can be advantageous if you do not require all the facilities of full-function processing. Examples of such applications include teller transactions in banking and point-of-sale transactions (inventory update) in retail marketing. Fast Path input and output messages use the expedited message handler (EMH) and bypass message queuing and the priority-scheduling process.

Extended Terminal Option (ETO)

IMS Extended Terminal Option (ETO) provides dynamic terminal and local and remote logical terminal (LTERM) support for IMS TM.

Definition: A *logical terminal (LTERM)* is a user destination. For statically defined terminals, each LTERM is associated with a physical terminal. For ETO terminals, each LTERM is associated with a user, and is associated with a physical terminal only after a user has signed on to a physical terminal.

You can add terminal hardware and LTERMs (users) to the IMS without first defining them. ETO gives you the option of eliminating macro statements in the IMS system definition of VTAM terminals and LTERMs.

[9] You can also use z/OS ATBxxx call services. For information on these call services, see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*

ETO enhances the availability of your IMS by eliminating the need for you to bring the system down in order to add terminals and LTERMs.

ETO also enhances security for the IMS TM user by associating all output with a specific user, instead of with a device. ETO requires the user to sign on.

ETO reduces the IMS system definition time for those systems where the terminal network is defined dynamically.

Intersystem Communication (ISC)

You can exchange data between your IMS and other external subsystems.

Definition: An *external subsystem* is a subsystem that provides a set of database resources that IMS can use, but does not control. Examples of external subsystems to which your IMS can connect include:

- Customer Information Control System (CICS)
- Other IMS systems
- User-written subsystems

The session that is created between IMS and an external subsystem is called an *application-to-application session*. The IMS that operates in this session uses a function called Intersystem Communication (ISC), which uses SNA protocols.

Message Format Service (MFS)

IMS Message Format Service (MFS) is a facility that formats messages to and from terminals, so that application programs need not deal with device-dependent data in input or output messages. An application program can format messages for different device types using a single set of editing logic, even when device input and output differ from device to device.

IMS MFS formats messages to and from user-written programs in remote controllers and subsystems so that application programs need not deal with the transmission-specific characteristics of the remote controller.

Multiple Systems Coupling (MSC)

Multiple Systems Coupling (MSC) enables you to enter transactions in one IMS and process them in another IMS. The responses from IMS can be returned to the terminals that entered the transactions, or to other terminals.

Related concepts

“Fast Path expedited message handler” on page 380
This topic briefly describes the Fast Path expedited message handler (EMH) and how it processes Fast Path messages.

“IMS messages and their scheduling” on page 386
An *IMS message* can be one of four types of data communication for which IMS controls processing: transactions, messages sent to LTERMs, IMS Commands, and DFSAPPC.

“Logical terminals (LTERMs)” on page 385
A *logical terminal (LTERM)* is a user destination. For statically defined terminals, each LTERM is associated with a physical terminal. For ETO terminals, each LTERM is associated with a user, and is associated with a physical terminal only after a user has signed on to a physical terminal.

“Message Format Service” on page 425

Message Format Service (MFS) is an IMS facility that formats messages to and from terminals, so that IMS application programs need not deal with device-specific characteristics in input or output messages.

Related tasks

"CPI Communications and APPC/IMS" on page 23
These topics introduce CPI-Communications and APPC/IMS. The topics discuss how CPI-Communications driven application programs function and how to administer APPC/IMS and use APPC/IMS with the CPI Communications interface to build CPI application programs.

"Extended Terminal Option (ETO)" on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

"Intersystem Communication (ISC)" on page 443
These topics introduce Intersystem Communication (ISC) and provide all the information you need to use ISC to connect IMS subsystems with other types of subsystems that support the ISC protocol.

"Multiple Systems Coupling (MSC)" on page 665
These topics introduce multiple systems coupling (MSC). You can use MSC to connect multiple IMS subsystems. These topics include an overview of MSC and the information you need to design, implement, and administer an MSC network.

The Data Communication Control (DCCTL) environment

The Data Communication Control (DCCTL) environment allows you to use the IMS Transaction Manager independently of the IMS Database Manager.

DCCTL provides improved system performance in terms of throughput, system availability, and integrity. DCCTL can coexist with current IMS application programs and installed terminals.

The following scenarios involving DCCTL do not require modifications to your existing environment:

- Application programs that access external database managers can use DCCTL without any modifications. For example, DCCTL can function as a front-end transaction manager for Db2 for z/OS. Db2 for z/OS subsystems do not require modifications in order to run with DCCTL.
- IMS exit routines and IMS application programs that access external subsystem resources in a DCCTL environment do not require modifications.
- Global resource management is the same in a DCCTL environment as it is in a DB/DC environment.

The following procedures might require modifications for a DCCTL environment:

- Operational procedures
- The general procedure that is produced by the system definition
- Application programs that contain a mixture of calls that access both external subsystems and IMS databases do require changes. DL/I calls result in a status code of AD.

Your existing system definition procedures support the generation of a DCCTL system. DCCTL executes with a collection of control blocks that identifies your data processing environment. These control blocks describe the system, data communication, and transaction manager components.

Using a DCCTL environment class system definition, you can generate a TM batch environment. Using TM batch, you can either take advantage of the IMS Batch Terminal Simulator (BTS) or access Db2 for z/OS systems. TM batch support allows only DBB and DL/I regions. It does not provide DL/I database capabilities.

Related reading: For more information on accessing Db2 for z/OS, see *DB2 for z/OS Application Programming and SQL Guide*.

Related concepts

DCCTL environment (System Administration)

Operating an IMS network

Operating a basic IMS network involves several tasks.

Operating tasks include:

- Establishing a communication session between a logical unit and IMS
- Sending data between a logical unit and IMS
- Terminating the session between a logical unit and IMS
- Restarting the session after a failure

Operating the network with APPC/IMS

APPC/IMS supports IMS commands for network operation, but the LU 6.2 device handles normal operations such as session startup, transaction initiation, and error handling that does not require master terminal operator (MTO) intervention.

Initiating a session with IMS

A session can be initiated by a logical unit, by the VTAM network operator, by the IMS master terminal operator (MTO), automatically by VTAM, or by IMS itself. After a logical unit connects to IMS, it remains connected until one of the following actions occurs:

- The logical unit itself requests disconnection.
- The IMS MTO requests disconnection.
- Another VTAM application program requests connection to the terminal.
- IMS, VTAM, NCP, the logical unit, or the entire network is stopped.

After the physical connection between the controller and VTAM is established, an LU-to-LU session is initiated. The LU that is requesting the session informs VTAM that it wants to communicate with IMS. VTAM notifies IMS of the request through the VTAM Logon exit routine. IMS indicates that it will accept the request, and VTAM then logically connects the LU to IMS. A session is required before communication between the LU and IMS can be accomplished. To open a session, all nodes in the communication path (IMS, NCP, line, and station) must be in active status. After a session is established, VTAM directs all data between the logical unit and IMS.

IMS also supports SNA communication links in an Extended Recovery Facility (XRF) complex.

Logging on and signing on to IMS

The following definitions apply to logging onto IMS and signing onto IMS:

Definitions:

- *Logging on* to a terminal establishes a session with IMS for that terminal.
- *Signing on* to a terminal identifies a user to IMS.

Logging off and signing off from IMS

The following definitions apply to logging off from IMS and signing off from IMS:

Definitions:

- *Logging off* from a terminal ends a session with IMS for that terminal.
- *Signing off* from a terminal ends an identification of a user to IMS.

Related concepts

Extended Recovery Facility Overview (System Administration)

The shared-queues environment

Operating in a shared-queues environment allows multiple IMS systems in a sysplex environment to share IMS message queues and EMH message queues. The IMS systems work together as an IMSplex providing a single-image view of multiple IMS systems.

The shared-queues environment distributes processing loads between the IMS systems in the IMSplex. Transactions that are entered on one IMS can be made available on the shared queues to any other IMS that can process them. Results of these transactions are then returned to the initiating terminal. End users need not be aware of these activities; they view the processing as if they were operating a single-system.

Definitions:

- A *shared queue* is a collection of messages that are associated by the same queue name. A shared queue is managed by a Common Queue Server (CQS) and can be shared by CQS clients in a IMSplex.
- A *Common Queue Server* receives, maintains, and distributes data objects from a shared queue that resides in a coupling facility list structure for its client.
- A *CQS client* is an IMS DB/DC or DCCTL system that accesses shared queues through its own CQS.
- A *coupling facility* is a special, logical partition that provides high-speed caching, list processing, and locking functions in a sysplex environment.
- A *sysplex environment* is a set of z/OS systems that communicate and cooperate with one another through certain multisystem hardware components and software services in order to process workloads.
- An *IMSplex* is one or more IMS control regions, managers, or servers that work together as a unit. Typically, but not always, IMS systems in an IMSplex:
  - Share either databases or resources or message queues (or any combination)
  - Run in a z/OS Parallel Sysplex® environment
  - Include an IMS Common Service Layer (CSL)

In general, IMS handles messages in the following manner:

1. IMS systems register interest in those queues for which they are able to process messages.
2. When an IMS receives a message and places it on the shared queue, all IMS systems that have registered interest in that queue are notified.
3. One IMS retrieves the message and processes it.
4. The IMS that processes the message places a response on the queue.
5. The IMS that submitted the original message is notified that the response message was placed on the queue.
6. The IMS that submitted the original message sends the response message to the originating terminal.

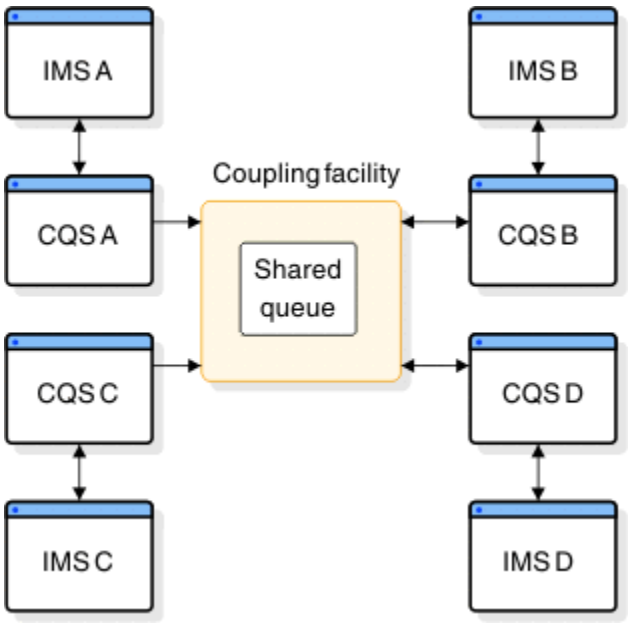


Figure 44. Basic shared-queues environment

Related reference

Queue types (System Administration)

Benefits of using shared queues

The major benefits of operating in a shared-queues environment are automatic workload balancing, incremental growth, and increased reliability.

The following list explains how shared queues provides these benefits:

Automatic workload balancing
A message that is placed on a shared queue can be processed by any participating IMS that is available to process work.

Incremental growth
You can add new IMS systems as workload increases.

Increased reliability
If one IMS fails, work that is placed on a shared queue can still be processed by other IMS systems.

Recommendations:

- Use generic resource groups with shared queues.
- Share all data in an IMSplex across the IMSplex.

Related concepts

"Balancing sessions with generic resources" on page 377
If you are operating in a sysplex environment and have multiple IMS systems, you can initiate a session by using the name of a generic resource group. VTAM balances the sessions among generic resource members in a generic resource group.

Data sharing in IMS environments (System Administration)

Required components of a shared-queues environment

Shared-queues processing requires a number of different components.

Although you can operate many different configurations of a shared-queues environment, the required components of shared-queues processing, shown in Figure 45 on page 376, include:

Common Queue Server (CQS)
One CQS is required for each client, though multiple IMS systems can share a CQS. Each CQS accesses the shared queues, which reside on coupling facility list structures.

CQS client
One or more IMS DB/DC or DCCTL subsystems that can access the shared queues using CQS client requests.

z/OS coupling facility list structures
A type of coupling facility structure that maintains the shared queues.

Definitions:

- A *list structure* is an area of storage in a coupling facility that enables multisystem applications in a sysplex environment to share information that is organized as a set of lists or queues. The list structure consists of a set of lists and an optional lock table.
- CQS maintains list structures in pairs, called *structure pairs*, with a primary list structure and an overflow list structure.
- The *primary list structure* contains the shared queues.
- The *overflow list structure*, if defined, contains shared queues that overflow after the primary list structure reaches a predefined threshold.

z/OS system log
One z/OS system log is used for each structure pair. CQS places recovery information about the work it has processed and about the list structure pair in the z/OS log streams. These log streams are then shared by all CQSs that access the list structure pair.

CQS checkpoint data set
One CQS checkpoint data set is maintained for each structure pair of each CQS. The CQS checkpoint data set contains CQS system checkpoint information.

CQS structure recovery data sets (SRDSs)
CQS maintains two SRDSs for each structure pair so that shared queues on the structures can be recovered. The SRDSs maintain structure checkpoint information for the shared queues.

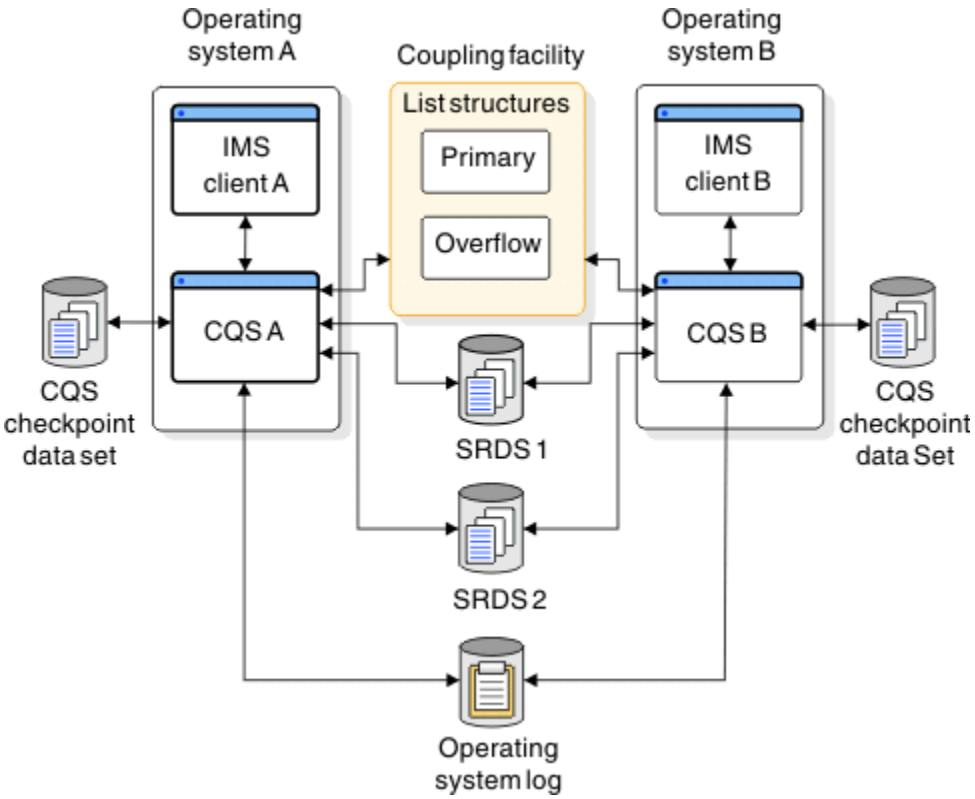


Figure 45. Components of a shared-queues environment

Related concepts

"Overview of the Common Queue Server" on page 376
The Common Queue Server (CQS) is an internal interface by which IMS communicates with shared message queues. You can use IMS commands to initiate CQS requests.

Writing a CQS client (System Programming APIs)

Overview of the Common Queue Server

The Common Queue Server (CQS) is an internal interface by which IMS communicates with shared message queues. You can use IMS commands to initiate CQS requests.

The CQS address space is started by the IMS.

CQS provides the following services:

- Notifies registered clients when work exists on the shared queues.
- Provides clients with an interface for accessing shared queues and CQS.
- Writes CQS system checkpoint information to a CQS checkpoint data set.
- Writes structure checkpoint information to an SRDS for recovery of a shared-queues list structure.
- Provides structure recovery and overflow processing for the shared-queues list structure.
- Drives the following CQS client exit routines:
  - The Client CQS Event exit routine notifies the client of system events, such as CQS abnormal termination and CQS restart completion.
  - The Client Structure Event exit routine notifies the client of structure events, such as structure copy, structure recovery, structure overflow, structure checkpoint, and structure resynchronization.
  - The Client Structure Inform exit routine notifies the client when work exists on the shared queues.
- Drives the following CQS user-supplied exit routines:

- The CQS Queue Overflow user-supplied exit routine allows the exit to approve or reject a queue that CQS selects for overflow.
- The CQS Initialization/Termination user-supplied exit routine is notified when CQS initializes and when CQS terminates after disconnecting from all structures (under normal termination conditions).
- The CQS Client Connection user-supplied exit routine participates in connecting clients to structures and in disconnecting clients from structures.
- The CQS Structure Statistics user-supplied exit routine gathers structure statistics during CQS system checkpoints.
- The CQS Structure Event user-supplied exit routine tracks structure activity, such as structure checkpoint, structure rebuild, structure overflow, and structure status change. It also tracks when CQS connects to a structure or disconnects from a structure.

- Provides the Log Print utility with sample JCL. By using the sample JCL, you can print log records from the z/OS log.

Related concepts

CQS administration (System Administration)

Enabling shared queues (System Administration)

Related reference

CQS client exit routines (Exit Routines)

Balancing sessions with generic resources

If you are operating in a sysplex environment and have multiple IMS systems, you can initiate a session by using the name of a generic resource group. VTAM balances the sessions among generic resource members in a generic resource group.

If you do not require the services of a specific IMS, initiate the session using a generic resource name, rather than the APPLID name of a specific IMS. VTAM Generic Resources is intended to run in a sysplex environment, but a sysplex environment is not required.

This topic gives an overview of the benefits and terminology of VTAM Generic Resources.

Benefits of generic resource groups

The benefits of using generic resource groups include:

Automatic session workload balancing
Using generic resources is complementary to using shared queues; generic resources distributes network traffic among multiple IMS systems, while shared queues distributes back-end application workload.

Single-image resources
You can use a single generic resource name to access multiple IMS systems, offering a single-system image while you are using the resources of many IMS systems.

Enhanced IMS system availability
In general, if one IMS fails, you can log on to another IMS in that generic resource group.

Exception: You might not be able to logon if the terminal has LOCAL status recovery mode.

Share global messages
You can obtain messages on shared queues from any IMS in the generic resource group.

Recommendation: Before attempting to obtain messages from a terminal that is in either conversation or fast-path response mode, you may need to re-logon to the system from which the input was originally submitted. If the status recovery mode is LOCAL, you must re-logon to the system from which the input was originally submitted. If the status recovery mode is GLOBAL or NONE, you can logon to any IMS within the generic resource group.

Generic resource group definitions

Definitions:

- A *generic resource group* is a set of IMS subsystems that have the same generic resource name, enabling VTAM to distribute terminal sessions among them.
- A *generic resource member* is an IMS subsystem that belongs to a generic resource group.
- A *generic resource name* is the common name by which VTAM knows all the IMS subsystems that belong to a generic resource group.
- An *APPLID name* is a unique application program name by which VTAM knows an IMS subsystem. In a generic resource group, VTAM uses the APPLID name to differentiate each IMS subsystem.
- In the context of generic resources, an *affinity* is an association between a VTAM logical unit and a specific IMS subsystem in a generic resource group.

Generic resource affinity

VTAM establishes an affinity between a terminal and a specific IMS in the following circumstances:

- For all sessions initiated by IMS
- When logging onto IMS using a generic resource name

Until the affinity is reset, subsequent logons resolve to the same IMS.

Affinity is automatically reset in one of two ways:

- By VTAM at session termination if VTAM-managed affinity is established for the session.
- By IMS at session termination if IMS-managed affinity is established for the session, unless the terminal has end-user significant status in the local IMS (status recovery mode of LOCAL).

Related concepts

Planning for VTAM generic resource groups (System Administration)

IMSplex terminal management

Using Resource Manager (RM) and a resource structure enhances your ability to manage IMSplex-wide TM resources and to share terminal-related information in a DB/DC or DCCTL environment.

When operating with the Common Service Layer (CSL), specifically RM, the resource structure provides a way to consistently define and maintain resources across the IMSplex. By sharing resource information throughout the IMSplex, you also gain transparency and state recovery for terminals and users.

This topic gives an overview of the benefits of using RM and a resource structure to manage your TM resources.

Note: If you have defined your IMSplex without a resource structure, terminal management is the same as it is for IMS systems running in local mode only.

Related concepts

CSL RM administration (System Administration)

CSL administration (System Administration)

Planning for Transaction Manager resources in an IMSplex (System Administration)

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

Benefits of managing resources with a resource structure

Using Resource Manager (RM) and a resource structure to manage IMS TM resources provides a number of benefits.

The benefits of using RM and a resource structure include:

Enforcement of resource type consistency
Prevents the same name being used for more than one resource type

Enforcement of resource name uniqueness
Ensures that certain resources can only be active one at a time within the IMSplex

Global callable services
Allows exit routines to obtain LTERM, node, and user resource information across the IMSplex

Terminal and user status recovery
Allows terminals and users to resume work without having affinity to any IMS

VTAM can manage Generic Resource affinities
Specifying GLOBAL as the status recovery mode allows VTAM to manage Generic Resource affinities instead of IMS. If an IMS system fails, VTAM can reassign terminals to another IMS system even if the terminals have affinity to the failed IMS system.

Retrieval of enablement values for global IMS functions across IMS cold starts
If you enable an IMS function globally by using the **UPDATE IMSFUNC** command and you use RM and a resource structure, the function enablement value is stored in a resource structure so that the value is retrieved across an IMS cold start.

Related concepts
How RM and the resource structure impact IMS activities (System Administration)

Related reference
DFSDCxxx member of the IMS PROCLIB data set (System Definition)

Shared TM resources

TM resource sharing is automatic when a resource structure is present in an IMSplex with Resource Manager (RM).

When sharing TM resources the following points apply:

- Each IMS connected to the resource structure can use the TM resources of all the other IMS systems connected to the resource structure.
- RM enforces resource name uniqueness among IMS systems connected to the resource structure.
- RM enforces resource type consistency among IMS systems connected to the resource structure.

If you need to disable TM resource sharing while maintaining all of the connections between the resource structure and the IMS systems in the IMSplex, specify STM=NO in the DFSDCxxx PROCLIB member.

Related reference
DFSDCxxx member of the IMS PROCLIB data set (System Definition)

Resource name uniqueness

When Resource Manager (RM) is active and a resource structure is defined, IMS automatically enforces name uniqueness for certain TM resources.

The TM resources for which IMS enforces name uniqueness include:

- VTAM LTERMs
- VTAM single-session nodes
- User IDs
- Users

You can disable the enforcement of resource name uniqueness, and TM resource sharing in general, by specifying STM=NO in the DFSDCxxx PROCLIB member.

Related reference
DFSDCxxx member of the IMS PROCLIB data set (System Definition)

Resource type consistency

IMS automatically enforces type consistency for TM resources when RM is active and a resource structure is defined in the coupling facility.

IMS validates the type of resources for the following message destinations:

- LTERMs
- LTERMs defined as APPC descriptors
- MSNAMEs
- CPI-C transactions
- non-CPI-C transactions

You can disable the enforcement of resource type consistency when a resource structure is present by specifying STM=NO in the DFSDCxxx PROCLIB member. After specifying STM=NO, resource type consistency is enforced only for non-CPI-C and CPI-C transactions.

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

Fast Path expedited message handler

This topic briefly describes the Fast Path expedited message handler (EMH) and how it processes Fast Path messages.

Fast Path message scheduling

Fast Path schedules input messages by associating them with a load balancing group.

Definition: A *load balancing group* is a group of Fast Path input messages that are ready for balanced processing by one or more copies of a Fast Path program. One load balancing group exists for each unique Fast Path message-driven application program.

The messages for each load balancing group are processed by the same Fast Path program. The EMH controls Fast Path messages by:

- Managing the complete execution of a message on a first-in-first-out basis
- Retaining the messages that are received in the control program's storage without using auxiliary storage or I/O operations
- Supporting multiple copies of programs for parallel scheduling
- Requiring that programs operate in a wait-for-input mode

Routing codes and balancing groups

IMS places input messages on message queues by using the transaction code, and attempts to process one queue until that queue is exhausted, within limits specified by system definition. Message processing is grouped for load balancing and synchronized for database integrity and recovery.

Definitions:

- A *message queue* is a data set on which messages are placed before being either processed by an application program or sent to a terminal.
- A *transaction code* is a one- to eight-character alphanumeric code that invokes an IMS message processing program.
- A *routing code* is a user-defined code that the EMH uses to enable transactions to be routed to programs within a load balancing group. A message is assigned to a balancing group by a routing code. The routing code can be the same as the transaction code, or it can be any other name you choose.

Incoming messages have preassigned routing codes, or a routing code can be dynamically set for the input message. You declare routing codes during system definition, and they are then associated with individual application programs.

Fast Path input edit/routing exit routine (DBFHAGU0)

All Fast Path-exclusive or Fast Path-potential messages are sent to the Fast Path Input Edit/Routing exit routine so that the application program can examine, change, or edit the input message in the input area.

Message buffering with a Fast Path-capable system

IMS buffer management uses a single EMH buffer when the input message has all of the following characteristics:

- Source terminal is VTAM
- Terminal is not enabled for front-end switch (FES)
- Input is single segment
- Input is non-MFS edited
- Input is not an IMS command
- Execution of at least one transaction is through Fast Path

An expedited message handler (EMH) buffer is dynamically allocated to the terminal when IMS receives the first message that has all of the characteristics listed. The EMH buffer remains allocated to the terminal for use with subsequent input messages (that have the same characteristics) until the user signs off, or until the session terminates. The buffer is then released to the EMH pool.

Fast Path EMH and shared queues

In a shared-queues environment that includes Fast Path, you have the option of sharing Fast Path EMH messages using an EMH queue (EMHQ). The EMHQ distributes the processing of EMH messages among multiple IMS systems.

If you do not intend to share EMH messages, you can delete the EMHQ statement from the DFSSQxxx PROCLIB member to prevent IMS from allocating the EMHQ structures and data sets.

Criteria for Fast Path application programs using EMH

Application programs that use EMH must conform to all of the following criteria:

- Each transaction must be initiated by a single-segment, response-type message.
- Each input message must require only a single-segment response message or no response.
- The input and output message length cannot exceed the EMH buffer size.
- ETO and LU 6.2 users cannot access main storage databases (MSDBs) with terminal-related keys.
- No IMS conversational processing is performed.
- Fast Path programs cannot act as automated operator programs or issue IMS commands.
- Fast Path transactions cannot be sent over any of the four types of multiple system coupling (MSC) physical links for processing in another IMS.

Related concepts

Enabling shared queues (System Administration)

Fast Path (System Administration)

Related reference

Fast Path Input Edit/Routing exit routine (DBFHAGU0) (Exit Routines)

Chapter 23. Planning the network

The following topics describe information that will help you plan your IMS network.

Planning for network administration

As an IMS network administrator, you should plan for each of the activities listed in the following table, which also lists where you can find additional information on each activity.

Table 71. Network administration activities

Activity	Related reading
Identifying the terminals and other devices required in the online system, verifying that IMS supports these devices, and identifying any incompatibilities.	For information on the terminals that IMS supports, the communication modes that IMS supports for each terminal, and the features for each terminal, control unit, or CPC, see Terminals and equipment supported by IMS 15 (Release Planning).
Establishing appropriate security features in the network	For information on establishing security, see “Planning for security” on page 401.
Determining message editing requirements	For information on editing and formatting messages, see Chapter 25, “Editing and formatting IMS messages,” on page 425.
Determining requirements for exit routine	For information on using exit routines to customize your IMS network, see Exit routines (Exit Routines).
Enforcing suitable naming conventions	For information on establishing naming conventions, see Establishing naming conventions (System Administration).
Defining the network at IMS system definition, and coordinating the system definitions for the host	For information on IMS network system definition, see Defining terminals with data communication macros (System Definition).
Defining ETO descriptors and MFS device characteristics table entries	For information on ETO and the MFS device characteristics table, see Chapter 5, “Administering the Extended Terminal Option,” on page 69.
Defining LU 6.2 descriptors to APPC/IMS	For information on APPC/IMS and LU 6.2, see Chapter 3, “Administering APPC/IMS and LU 6.2 devices,” on page 33.
Testing the readiness of the network	For information on testing the network, see Ensuring network readiness (System Administration).
Monitoring the performance of the network, and analyzing message loading, such as transaction routing and message queue size	For information on performance monitoring, see Performing capacity planning (System Administration).

Documenting network and terminal requirements

The size of your IMS installation and your terminal profile determine how you document your network requirements.

The terminal profile

© Copyright IBM Corp. 1974, 2020

Creating a terminal profile is the best approach for gathering the information you need for your system definition. Your terminal profiles also assist you in coordinating the installation arrangements. They help you describe, to the system programming staff, how each terminal is to be used.

Begin by examining the application program specifications or the hardware plans to identify the terminals that are to be used.

Terminals attached using VTAM

Terminals attached through VTAM are defined according to terminal type. Communication and attachment modes include:

- Binary synchronous communications (BSC)
- Synchronous data link control (SDLC)
- Local attachment (directly to a channel)

When you define your IMS or select an ETO logon descriptor, your choice of terminal type determines the communication protocol and the MFS formatting that IMS uses for that terminal. You can use more than one terminal type for a particular physical terminal; however, your choice must be compatible with the application programs that execute on the IMS.

IMS terminal network

An IMS network is characterized by physical terminals and other devices. IMS supports many terminals, including display devices, printers, and remote intelligent controllers.

IMS uses an access method, such as VTAM, to communicate with a physical terminal. Be sure to understand the difference between:

- VTAM's view of terminals that are connected to a host
- IMS's supervision of those connections

VTAM's responsibilities include:

- Receiving the input message and processing it
- Transmitting data
- Managing line and terminal communication status

IMS's responsibilities include:

- Monitoring the message traffic
- Routing messages to application programs
- Handling message storage, scheduling, and recording

Although IMS uses the network facilities of VTAM, it can also control devices that use BSAM.

VTAM is the preferred access method for IMS. You can use OTMA for non-VTAM networks.

To describe a physical terminal configuration to IMS, you code system definition macros, such as the TYPE macro, that describe the terminal attachments and their characteristics (such as communication type, features, or options). You also define ETO and LU 6.2 descriptors.

Terminal connections to IMS

IMS supports various modes of communication or attachment, usually using VTAM. Remote attachments are either switched or nonswitched.

Terminals that are attached by using VTAM are assigned a terminal type. The addresses of the lines and terminals that IMS uses are transparent to IMS, because they are not specified during IMS system definition. Instead, all VTAM resources have names assigned to them.

Definition: Each logical unit is assigned a *node name* that your installation defines.

The node name that is chosen for the VTAM definition is also used in IMS system definition and commands. Using VTAM enables IMS users to share network resources with other VTAM applications.

XRF terminal support

XRF supports all terminals that IMS supports. XRF terminal support is divided into three categories: class 1, class 2, and class 3.

Related concepts

“Terminals in an XRF complex” on page 406

XRF supports all terminals that your IMS users are currently using. XRF provides emergency support with a minimum of effort from you.

Extended Recovery Facility Overview (System Administration)

Logical terminals (LTERMs)

A *logical terminal (LTERM)* is a user destination. For statically defined terminals, each LTERM is associated with a physical terminal. For ETO terminals, each LTERM is associated with a user, and is associated with a physical terminal only after a user has signed on to a physical terminal.

Each logical terminal has an installation-defined name, called the *LTERM name*.

Reserve one LTERM and identify it as the IMS master terminal. This logical terminal is the control point for the online IMS. Commands associated with this LTERM start and stop the system, control the system resources, and display the status of those resources. The master terminal operator (MTO) can use the `/ASSIGN` command to alter the physical device that is associated with the LTERM. When ETO users sign on, altering the physical device that is associated with the LTERM is automatic. The device is identified by either the node name or the line number with the physical terminal identifier.

When a user enters a transaction, the logical terminal name is associated with the input message, and input messages are queued by transaction code. The output message queue designation is actually the LTERM name itself. Although this name is often the same for output as for input, an application program can cause the output to be directed to an alternate LTERM. An input terminal can also specify an LTERM name as the destination of a message. A message switching (input) edit routine can append the current input LTERM name to the message.

Related concepts

“Designing logical terminal networks” on page 390

The IMS system definition describes the characteristics and relationship of communication lines, static terminals, and logical terminals (LTERMs).

Related reference

Transaction Manager exit routines (Exit Routines)

APPC/IMS and LU 6.2 terminal support

APPC/IMS does not use an LTERM for input. APPC/IMS provides two facilities (that are similar to LTERMs) for determining output destinations: LU 6.2 descriptors and side information.

LU 6.2 descriptors allow an IMS LTERM name to define an LU 6.2 destination (specifying LUNAME, TPNAME, and other LU 6.2 destination characteristics). These LU 6.2 LTERMs can be used the same way as regular LTERMs in IMS application programs.

Side information contains system-defined values provided by CPI Communications (CPI-C), and supplies LU 6.2 destination information. For APPC/IMS to establish a conversation with a partner program, CPI-C requires initialization information, such as the name of the partner program and the name of the LU at the partner's node. CPI-C provides a way to use system-defined values for these required fields. These system-defined values are specified in the side information, and are accessed by a symbolic destination name. The symbolic destination corresponds to an entry in the side information containing the partner_LU_name, the mode_name, and the TP_name.

Related reading: For more information on side information, symbolic destination names, and the entries in the side information, see *z/OS MVS Planning: APPC/MVS Management*.

IMS messages and their scheduling

An *IMS message* can be one of four types of data communication for which IMS controls processing: transactions, messages sent to LTERMs, IMS Commands, and DFSAPPC.

Transactions
Transactions are input messages that are destined for processing by application programs. Transactions are identified by a one- to eight-character *transaction code*. Transactions can be entered at terminals or generated by application programs. In a multiple-systems environment, a transaction can originate in a remote IMS, or in another subsystem (like Db2 for z/OS).

Messages sent to LTERMs
Messages that are sent to LTERMs are identified by LTERM names. These messages can be either of the following types:

- Output messages sent from application programs to communicate with a logical terminal. They usually acknowledge or give results of work, and are sent to the originating terminal. Application programs can also send output to logical terminals other than the one that originated the input message.
- Input messages whose destination is a logical terminal. This message type is known as a *terminal-to-terminal message switch*. The text of the message becomes the output message to the destination LTERM.

IMS commands
IMS commands are entered by terminal users who request that IMS display or alter the status of one or more IMS resources. Most commands originate from the master terminal operator (MTO), but other terminals, as well as application programs, can also enter them.

DFSAPPC
DFSAPPC is an IMS service that is used for exchanging messages between LU 6.2 devices, or between any combination of LU 6.2 and non-LU 6.2 devices. Messages are delivered by allocating a new conversation with the designated LU 6.2 destination device. If the allocated conversation fails, the output is stored for future delivery. Messages are held on the IMS message queue until they can be successfully delivered.

Message queuing

In IMS systems that do not used shared queues, all input and output messages are queued in IMS-controlled virtual storage. Most command output is queued in virtual storage. Application programs handle messages serially if SERIAL=YES is specified on the TRANSACT macro.

If virtual storage is exceeded, messages are saved on direct-access storage. In this way, messages can be received as input or saved for output, although the resources that are necessary to process them might not be immediately available.

The maximum number of concurrent terminal I/O requests and the amount of virtual storage to hold the messages are specified during IMS system definition. Both variables can be altered when IMS is started.

In a shared queues environment, input and output messages are queued to a queue structure in a coupling facility. The shared queues environment is documented in *IMS Version 15 System Administration*.

Message segments

When a terminal is used to enter data into the IMS online system, the input can consist of either single segments or multiple segments. A sequence of several segments might be required to specify a message that can be interpreted by an application program. The device operation determines the end-of-segment.

For input, detection of the end-of-message condition by IMS indicates that a complete message has been received.

When the first end-of-segment condition is detected, IMS examines the beginning of the segment to find a destination. If a destination is declared to be a single-segment message, the end-of-segment signals the end-of-message.

After editing, the first (or only) segment of a message has the following structure:

```
   LL       ZZ      DEST-CODE     b        MMMM
2 bytes  2 bytes  1 to 8 bytes  1 byte   message text
```

These parameters are explained in the following table.

Table 72. Message segment format

Message segment part	Bytes	Explanation
LL	2	Total length of segment including LL and ZZ parts
ZZ	2	Reserved halfword
DEST-CODE	1 to 8	Destination code (usually a transaction code)
MMMM	Varies	Message text

Each input message is uniquely identified by its destination code. The destination is normally a transaction code, but it can also be an IMS command or LTERM for message switching.

Invalid destinations

Even when the destination code meets normal resource naming conventions, the input destination can be incorrect:

- If ETO is not active, destinations that cannot be identified in IMS are considered invalid, and the input message is rejected.
- If ETO is active, IMS assumes these destinations are LTERM names and creates dynamic user structures. If the destination is invalid, a *dead-letter queue* is created. Several commands are provided, including the /DISPLAY command, for the MTO to process the dead-letter queues.

Transaction codes

The transaction code has optional attributes that affect the processing program's scheduling eligibility by the IMS control program.

Application programs are defined in separate but related macro instructions. The application program that is designated to process a particular transaction code is considered by IMS to be another transaction attribute. An application program might process several transaction codes, but a transaction code can be associated with only one program.

Message scheduling

Definition: *Message scheduling* is the process by which a completely received input transaction is united with its associated application program for processing.

Some of the factors that affect the message scheduling process are:

- The variable attributes associated with the transaction code
- The number and relative importance of other transaction codes

- The number of received messages that are not yet processed
- The intent of associated application programs toward the data to be processed
- The amount of currently available space in control-block storage pools and buffers

In addition, each of the following activities affect the message scheduling process:

- Selecting system definition options
- Designing and using databases
- Specifying buffer sizes
- Declaring and selecting transaction code priorities

By properly combining these activities, you can manipulate the sequence in which messages are processed and enhance system performance.

CPI-C transactions

IMS resources are made available to the CPI Communications driven application program when the application program issues the APSB (Allocate_PSB) call. The CPI Communications driven application program can use the SAA Resource Recovery Commit (SRRCMIT) and Backout (SRRBACK) calls to initiate an IMS synchronization point or backout. CPI Communications driven application programs should use the SAA Resource Recovery calls to initiate an IMS sync point prior to program termination.

Definition: A *synchronization point* (also referred to as a *sync point* throughout this manual) is an occurrence within a program or subsystem wherein resources are committed and a reference point is established for subsequent restart, if necessary.

Recommendation: Define CPI transactions with a different message class from that used for non-CPI transactions.

Related concepts

DFSAPPC message switch (Application Programming)

Performing capacity planning (System Administration)

Related tasks

“Asynchronous output to an invalid destination” on page 102
IMS refers to data that cannot be delivered as "dead letter".

“CPI Communications and APPC/IMS” on page 23
These topics introduce CPI-Communications and APPC/IMS. The topics discuss how CPI-Communications driven application programs function and how to administer APPC/IMS and use APPC/IMS with the CPI Communications interface to build CPI application programs.

Message flow within the IMS online system

This topic describes the events that take place as a message that is entered at a terminal flows through an IMS online system. Although some exceptions to the order of events and processing exist, the majority of transactions are processed this way.

IMS provides two ways to customize processing: basic edit (the default) and MFS.

Message flow from terminal to program

The initial entry from a terminal is a message that is processed asynchronously by IMS. After the message is received from the terminal, the message is formatted using basic edit, MFS, or exit routines.

The IMS message scheduler then uses the status of the message queues and a prioritizing algorithm in order to select the next transaction to schedule. When the program is selected for scheduling in the dependent region, the first segment of the message is made available to the program.

Message flow from program to output terminal

While the application program is executing, it can use DL/I calls to access databases. The data communication support of DL/I is used by the program to request messages, using the GU function for the first message segment and the GN function to retrieve subsequent message segments. The program can then send a response to the entering terminal by inserting the reply to the I/O PCB. The entering terminal's LTERM is the I/O PCB after a successful GU call.

If the message is handled by MFS, it can be transformed from the program output format to the device output format. Otherwise, the message is passed unchanged to the device.

When the output message has been completely received by a terminal or by a program, it is dequeued. Any failure in message delivery causes IMS to keep the recoverable message queued for a later delivery.

Message flow from program to alternate destination

The application program can also send output to an alternate destination. Using an alternate PCB, the program can insert a message to another LTERM. In this case, the message handling is the same, but the message queue on which it is placed has a different LTERM name.

Message flow from program to program

The application program can also use the alternate PCB to send a message to a program; that is, it can generate a secondary transaction that is placed on a message queue. The processing of the secondary transaction is dependent on the selection of the transaction by the scheduling algorithm.

Related tasks

"Editing and formatting IMS messages" on page 425
IMS uses two methods to edit and format messages to and from terminals: Message Format Service (MFS) and basic edit routines.

Conversational transactions

Conversational transaction processing allows you to retain message continuity from a given terminal, even when the program that processes the conversation is not retained in storage throughout that conversation.

When the transaction is defined as conversational, the application program can use a scratchpad area (SPA) in order to interrelate messages from a given terminal.

The *scratch pad area (SPA)* is a work area used in conversational processing in order to retain information from an application program across executions of the program. A unique SPA is created for each conversation.

Contents of the SPA

Typical contents of the SPA are data items entered at the terminal and data that is retrieved from databases to be saved between iterations of the conversation. To simplify operator procedure and application design, database updates are retained in the SPA until the last iteration of the conversation.

Any subsequent data entry from a terminal that is already operating in conversational mode causes a message processing program (MPP) to receive both the contents of the SPA and the input terminal data. Each input message is considered an individual unit of work. Each interaction can be with a different program, although the same program is typically used.

Message processing for conversational transactions

When the message is a conversational transaction, the following sequence of events occurs:

- IMS removes the transaction code and places it at the beginning of a message segment. The message segment is equal in length to the SPA that was defined for this transaction during system definition. This is the first segment of the input message that is made available to the program. The second through the *n*th segments from the terminal, minus the transaction code, become the remainder of the message that is presented to the application program.
- When the conversational program has prepared its reply, it inserts the SPA to IMS. The program then inserts the actual text of the reply as segments of an output message.
- IMS saves the SPA and routes the message to the input LTERM.
- If the SPA insert specified that another program is to continue the same conversation, the total reply (including the SPA) is retained on the message queue as input to the next program. This program then receives the message in a similar form.
- A conversational program must be scheduled for each input exchange. The other processing continues while the operator at the input terminal examines the reply and prepares new input messages.
- To terminate a conversation, the program places blanks in the transaction code field of the SPA, and inserts the SPA to IMS.
- The conversation can also be terminated if the transaction code in the SPA is replaced by any nonconversational program's transaction code, and the SPA is inserted to IMS. After the next terminal input, IMS routes that message to the other program's queue in the normal way.

ETO conversations

ETO conversations maintain associations with users, rather than with terminals. The conversation is associated with the terminal only while the user is signed on. The conversation can be restarted on any other ETO terminal.

Program switches for conversational programs

A conversational program can use a deferred program switch or an immediate program switch.

Definitions:

- During a *deferred program switch*, the program responds to the originating terminal but causes the next input from the terminal to go to another conversational program.
- During an *immediate program switch*, the program passes the SPA (and, optionally, a message) to another conversational program without responding to the originating terminal. In this case, it is the next program's responsibility to respond to the originating terminal.

Related tasks

“Extended Terminal Option (ETO)” on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

Message switches

A *message switch* occurs when an input message specifies an LTERM name instead of a transaction code. IMS formats the message and inserts it into the output message queue for that LTERM.

Designing logical terminal networks

The IMS system definition describes the characteristics and relationship of communication lines, static terminals, and logical terminals (LTERMs).

For a nonswitched terminal, the relationship between a physical terminal and a logical terminal within IMS is a static relationship defined during system definition.

When only one user operates a physical terminal, only one logical terminal is associated with that physical terminal. If multiple users operate a physical terminal, the terminal is associated with many logical terminals.

You can structure the IMS system definition so that a separate logical terminal is assigned for each user of a particular static terminal.

Definitions: The information in this topic uses *physical terminal* to represent a*node*, for VTAM devices.

Logical terminals can be assigned to physical terminals for both input and output. When a logical terminal is assigned to a physical terminal for output, all messages that are sent to that logical terminal are transmitted to its associated physical terminal. More than one logical terminal can be assigned to a given physical terminal for output. Only one physical terminal can receive the output for a given logical terminal. The relationship between physical and logical terminals for output is shown in the following figure.

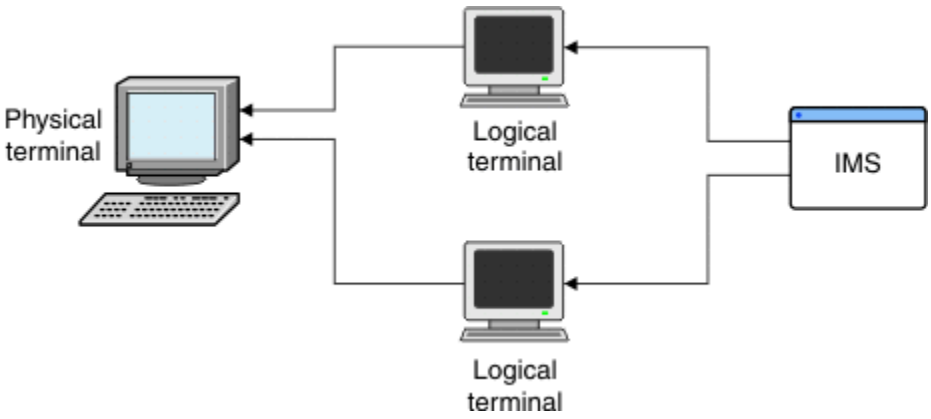


Figure 46. Relationship between physical and logical terminals for output from IMS

Logical-terminal chains

When a logical terminal is assigned to a physical terminal for input, any message that is entered from the physical terminal is considered to have originated at the logical terminal. When more than one logical terminal is assigned to a physical terminal for input, a chain of input logical terminals is formed. Any input from the physical terminal is considered to have originated at the first logical terminal on the chain.

The following figure shows the relationship between physical and logical terminals for input.

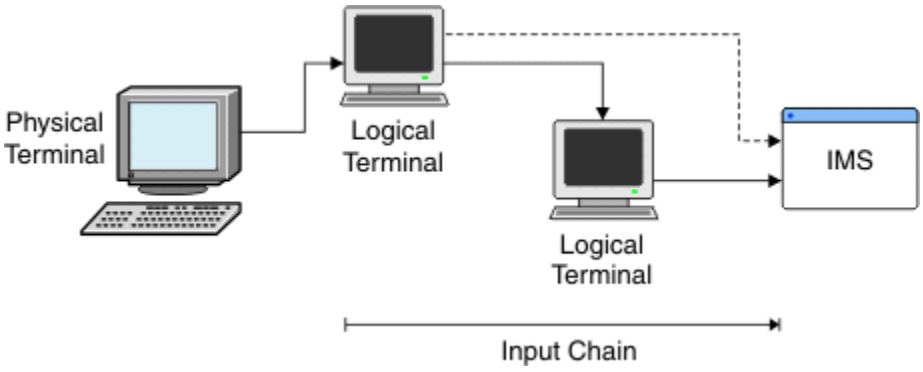


Figure 47. Relationship between physical and logical terminals for input to IMS

If the first logical terminal is not allowed to transmit a message (for example if it is not authorized or it is a stopped logical terminal), all logical terminals on the input chain are interrogated in chain sequence for their ability to transmit messages. If the physical terminal is a Finance, SLU 1, SLU P, or LU 6.1, only the logical terminals associated with the input component are scanned. The first appropriate logical terminal found is considered the originator of the message. If no appropriate logical terminal is found, the message is rejected with an error message.

Logical-terminal queues

Using a message queue for received input messages or for pending output messages enables an application program to be independent of the arrival time and message transmission. Because this message queue is associated with a logical terminal, rather than with a physical terminal, the message queue can be moved from device to device, independent of the application program. The message queue can even be moved among device classes.

Logical terminals provide stability for application programs. Regardless of how the physical terminal network changes, the application programs remain insensitive to these changes.

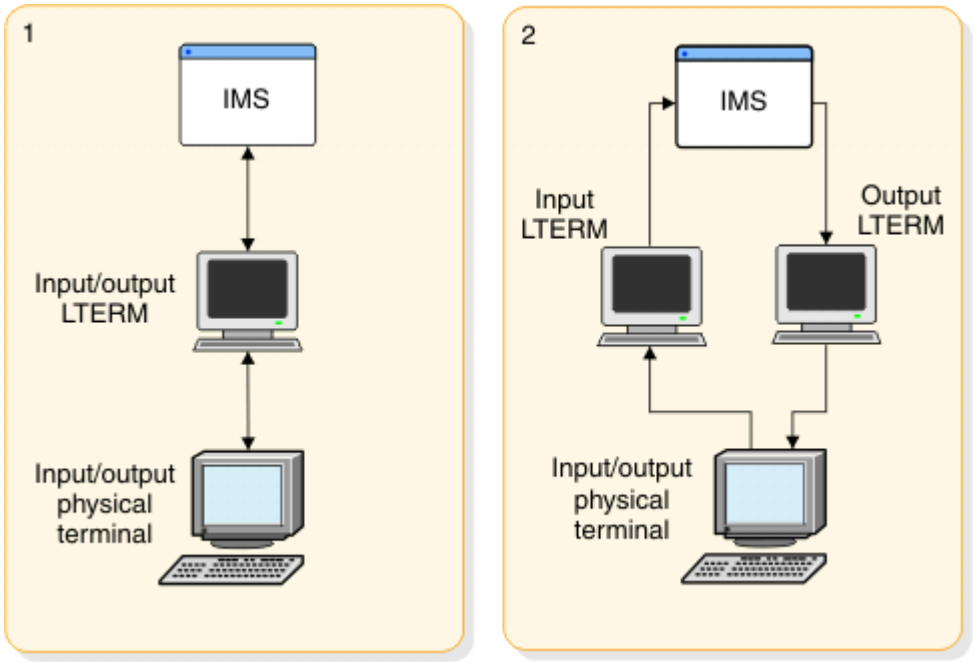
The application program interface to a logical terminal is conceptually the same as that for the database system:

- GU calls retrieve message segments from a queue.
- ISRT calls insert message segments to a queue.

Separating input and output devices

In certain application programs, it might be necessary to associate a different physical device for output than the one that is used for input. If the physical terminal type is an input-only device and output is required, a different device must be associated for output.

IMS system definition and commands support assignment of output devices that are different from the input device. For example, the application program that is processing a message from a display might need to send some of the output to a printer. The possible physical-to-logical relationships are shown in the following figure.



Notes to figure regarding applications

1. Normal assignment of one or more logical terminals or physical terminals. Output is sent to input terminal. Application is insensitive.
2. Application uses specific logical terminal for output. Application is insensitive to input.

Figure 48. Possible physical-to-logical terminal relationships

Logical and physical terminal relationships

The following figure shows the communication flow between a terminal user, a physical terminal, a communication line, and a logical terminal in a nonswitched communications network.

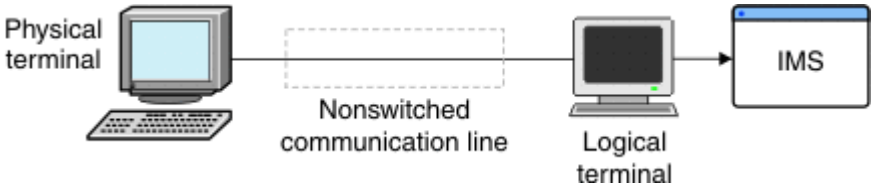


Figure 49. A nonswitched communications network

IMS system definition describes the characteristics and relationship of physical terminals, communication lines, and logical terminals. On a nonswitched communication line, the relationship between a physical terminal at one end and a logical terminal within IMS at the other end is a stable relationship defined during system definition or during signon for ETO terminals.

Except for LU 6.1, terminals using VTAM have physical-to-logical terminal (PTERM-to-LTERM) relationship as a terminal on nonswitched line.

The relationship that is established between a physical terminal and one or more logical terminals at system definition can be changed using commands or by creating a new system definition. The **/ASSIGN** command dynamically changes logical and physical relationships, and it is usually executable only from the master terminal.

Master terminal

The master terminal is the central point of control for IMS. If a VTAM master terminal is defined, IMS establishes a connection with it during startup.

Recommendation: Although several device types are supported, a VTAM SLU 2 device is the preferred type for the master terminal.

Restriction: Neither an ETO terminal nor an LU 6.2 terminal can be defined as the primary or secondary master terminal.

The master terminal operator (MTO) must be familiar with all aspects of operating the system. The physical location of the master terminal in relation to the computer console is important. If, for security reasons, the MTO and master terminal are not close together, telephone communication should be provided.

Master terminals in an XRF complex

In an XRF complex, each IMS must have its own master terminal, and it can have a secondary master terminal.

Typically, the secondary master terminal is a printer (output-only device type). System messages for the active and alternate subsystems go to their respective master terminals.

During a takeover, the console operators must know the status of the takeover on both systems. The operator at the alternate subsystem must know when I/O prevention in the failed active subsystem is complete; terminal switching of class-3 terminals and failed class-2 switch attempts must be handled.

Related concepts

Choosing the IMS master terminal (System Definition)

NTO terminals

When using the NTO licensed product, VTAM can support some start-stop and teletype devices, known as *NTO devices*.

The receipt of a message can be confirmed by using transaction-response mode to "lock" the keyboard following the input entry. When IMS responds with available output to a particular terminal for a specific transaction, it "unlocks" the keyboard and sends the data.

Even if the NTO devices are used primarily for inquiry transactions, the transactions can be made recoverable. Because input is not acknowledged, the application program should indicate in the output response the inquiry transaction that caused it.

Both input and output editing routines (which are provided by your installation and are entered after MFS processing) can be made available for transactions using NTO devices.

Because NTO uses VTAM, IMS is not aware of whether a terminal user is logging on to a session or is logging off. Therefore, existing session status might cause problems when one terminal user ends a session and another terminal user resumes the session. This problem can be minimized by defining all NTO devices as Response mode.

To avoid the problem of continuing session status, a terminal user can clear any preset transaction codes that were previously set using **/SET** commands by entering the **/RESET** command either:

- Before issuing the **/RCLSDST** command when ending an NTO session
- Before issuing the **/SIGN ON** command when starting an NTO session

A terminal user who enters **/RCLSDST** or the MTO who enters **/CLSDST** terminates a session but does not reset conditions that were set during that session.

Related concepts

How to communicate with IMS from an NTO device (Operations and Automation)

Resource modes and states

IMS keeps three types of status information for the terminals in the network.

The types of status information that IMS keeps are:

- The mode of operation
- The state of an inoperable terminal (such as temporarily unusable)
- The recovery status

Terminal and user operating modes

A terminal can be in more than one mode at the same time.

Possible modes for a terminal or user include:

Response Mode

Response mode is established through static terminals, ETO users, or transaction specifications. Response mode can be used with full function or Fast Path processing. In response mode, after the entry of an input message, the terminal is locked until a reply is received, and no additional incoming data is accepted.

Definitions:

- For operator-driven terminals like 3270s, *locked* means that the keyboard is locked.
- For programmable terminals like LUPs, *locked* means that IMS delays the acknowledgment of the input message until the output message is ready to be sent.

For LU 6.2, the originator of the transaction must issue a **Receive** command to get the response message (synchronous output). Failure to do so is treated as a protocol violation. Depending on the system, the application program could fail and IMS could send an error message.

Terminal-response mode is terminated when the response has been sent and dequeued. If Fast Path is used, terminal-response mode is automatically continued following an IMS failure, when a static terminal logs off, or when an ETO user signs off. If full-function operation is used, terminal-response mode is terminated when IMS is restarted, when a static terminal logs off, or when an ETO user signs off, unless the terminal or user is defined with both SRMDEF=LOCAL and RCVYRESP=YES, in which case the full-function response mode is recoverable after either a terminal logoff or a user signoff.

The ETO user, not the terminal, is in response mode. If the user signs off and Fast Path is used, response mode is recovered for the user, but the terminal is no longer in response mode. The user cannot enter another transaction until the response to the first transaction is received.

Conversation Mode

After you enter a transaction that is defined using the SPA= parameter on the TRANSACT macro, the terminal is in conversation mode. While the terminal is in conversation mode, other transactions cannot be entered from that terminal. However, the terminal is not locked, in the same sense that response mode locks the terminal. The terminal remains in conversation mode until the conversation is terminated, when the message has been sent and dequeued, and the application program has placed blanks in the transaction code field in the scratch pad area (SPA).

A conversation can abnormally terminate under the following conditions:

- When an application program abends
- When the IMS MTO issues an /EXIT command, a /START NODE command, or a /START USER command
- When an inconsistent definition exists in the application program between IMS and MSC

For LU 6.2, all iterations of the IMS conversation must use the same LU 6.2 conversation; each iteration of the IMS conversation is demarcated by using the LU 6.2 CMPTR (Prepare_To_Receive) call. If the LU 6.2 conversation ends prior to the end of the IMS conversation, the IMS conversation is abnormally terminated.

The ETO user, not the terminal, is in conversation mode.

Exclusive Mode

A terminal is placed in exclusive mode when the /EXCLUSIVE command is issued. The Exclusive mode:

- Restricts the output received by the terminal.
- Remains with ETO users after they sign off. ETO users that sign off while they are in exclusive mode remain in exclusive mode when they sign on the next time.
- Terminates with an /END or /START NODE command.

Lock Mode

Lock mode prevents a terminal from sending and receiving messages. A terminal, node, or logical terminal (LTERM) is placed in lock mode when the operator issues the /LOCK command. Lock mode is reset by issuing the /UNLOCK command.

Test Mode

Test mode ensures that any input message entered into a terminal is transmitted back to the terminal. A node is placed in test mode by the /TEST command. Test mode is reset by an /END command or a /START command. Test mode is not significant and is not carried across restart and ETO signoff and signon.

Related tasks

“Extended Terminal Option (ETO)” on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

Related reference

IMS commands (Commands)

Terminal and user states

Several terminal and user states render the terminal or user inoperable.

The states in which a terminal is inoperable include:

Stop State
The stopped state prevents the delivery of any output queued on a logical terminal that is associated with a physical terminal.

The /STOP NODE command results in the termination of the session between IMS and the node. This termination occurs immediately for most devices but only at the end of the message for 3270 devices and SLU 2 devices. The /STOP NODE command also prevents a new session until a /START command or /RSTART command is issued.

The /STOP command, the /PSTOP command, and the /MONITOR command also cause a terminal to enter the stopped state. This state is reset by the /START command or /RSTART command. The /STOP NODE or USER command allows ETO users to retain their status after signoff or logoff.

QERROR State
A logical terminal is placed in a stopped state if an I/O error is encountered while attempting to read from or write to a message queue. This condition is reset when the MTO operator issues a /START command.

QLOCK State
A logical terminal is prevented from sending any additional output until the locked state is reset by a specific request received in the LU 6.1 session. The LTERM is also prevented from receiving input that might create additional output. This condition is reset when the MTO issues a /START command. A /PSTOP or /PURGE command is ignored for LTERMs that are in QLOCK state.

INOP State
The physical terminal is considered inoperable by IMS device support whenever an error is detected and put in the INOP state. All logical terminals associated with this physical terminal are also considered inoperable. The /START or /RSTART command resets the inoperable condition.

COMPINOP State
Component inoperative can be set in one of two ways:

- When an error is detected that is isolated to a component of the terminal
- When the /COMPT or /RCOMPT command is issued for terminals that are defined to VTAM

All logical terminals associated with this component are ineligible for message output. The component inoperative state is reset when the operator issues a /START LINE PTERM command, a /START NODE command, another /COMPT command, or a /RCOMPT command. Special signals from the device, such as device end from a 3270 device or the "component available" status from a SLU 1 can also cause a reset.

PAGE, SCREEN, and COMPONENT PROTECTION State
This is a state supported for video terminals, SLU P, Finance, and LU 6.1 devices. Logical terminals associated with these physical terminals are not eligible for output selection.

SNA QUIESCE State
When IMS sends output messages to a VTAM programmable node and the node wants to stop receiving, the node signals IMS to stop transmissions after an end-of-chain has been sent. IMS does not send any more output to the terminal until the terminal sends the SNA release-quiesce (RELQ) command.

Related concepts

3270 terminal screen protection (Operations and Automation)

Related tasks

“Suspending output from IMS” on page 929
If the controller application program does not want or cannot receive any more output from IMS, the program can send the VTAM quiesce-at-end-of-chain (QEC) command to IMS. IMS returns a DR1 and the

VTAM quiesce-complete (QC) command. IMS does not send any more output to the workstation until it receives the VTAM release-quiesce (RELQ) command.

"Extended Terminal Option (ETO)" on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

Resource status recovery

Resource status recovery defines how a terminal or user is recovered.

Recovery information can be shared using RM; therefore a resource can be recovered without using the IMS log records. In an IMS failure, a terminal or user can resume work without having affinity to the failed system or having to wait for IMS restart. You can also choose to recover resources only on a local system, or to delete the resource status.

This topic defines the resource status classifications and recovery modes. It then describes how recovery mode status is used with Fast Path and XRF.

Resource status classification

IMS classifies resource status to determine how much information needs to be stored in the resource structure or in local log records.

Nonrecoverable Status
Status only exists when the resource is active. Status is deleted when the resource becomes inactive and is not recovered at terminal logoff, user signoff, or IMS restart.

Recoverable Status
Status is recovered, but does not prevent the resource from being deleted across signoff, logoff, or IMS restart.

Significant Status
Status is recovered and the resource is not deleted across signoff, logoff, or IMS restart. Where the status is maintained depends on whether the status is command or end-user.

Command Significant Status
Status relates to the resource command, such as STOP, TRACE, and MFSTEST. Status recovery is always maintained globally by RM in the resource structure, if defined. Status is unaffected by status recovery mode. Not all statuses set by commands are significant.

End-User Significant Status
Status relates to resource work: conversation, STSN, and Fast Path. The status frequently changes, which can affect performance. Therefore, you can specify the status recovery mode as either GLOBAL, LOCAL, or NONE.

Related reference
"Status recovery of TM resources" on page 400
IMS saves recoverable statuses for LTERMs, nodes, users, and user IDs. This topic lists the statuses saved for each.

Status recovery mode for end-user significant status

The status recovery mode defines the scope and location of recovery for resources with End-User Significant Status.

You can set the default mode for each IMS, which is used for all resources unless overridden during a logon or signon. You can then set what End-User Significant Status to recover.

The following list describes how to set the status recovery modes for TM resources:

- Specify IMS defaults in DFSDCxxx.
- Modify user descriptors.
- Set the DFSLGNX0 override for static terminal logon and dynamic STSN terminal logon.

- Set DFSSGNX0 override for dynamic non-STSN user signon.

The following list describes the three recovery modes for End-User Significant Status.

GLOBAL Status Recovery Mode

All recoverable status is saved locally in the IMS record logs, but uses RM to recover the status instead of the logs. Status is restored at the next logon or signon and is available to any IMS in the IMSplex. When the resource becomes active, status is copied to the local system. When the resource becomes inactive, status is deleted from the local system.

RM, a coupling facility resource structure, and shared queues are required. If you are sharing queues and have a resource structure, GLOBAL is the default. The default is overridden by DFSDCxxx, user descriptors, or logon and sign-on exit routines.

LOCAL Status Recovery Mode

All recoverable status is saved locally in the local control blocks and log records. Status is restored at the next logon or signon and is only available to the same IMS that the user or node was accessing. Additionally, the user or node can only access the IMS where the local status is. This affinity is called *RM affinity*, which is enforced when End-User Significant Status exists. With RM affinity, IMS does not allow a terminal or user to log on to or sign on to an IMS if RM indicates that the user or terminal has RM affinity to another IMS. This affinity occurs because end-user significant status (conversation, STSN, or Fast Path) is being recovered on another IMS.

When the IMS with the RM affinity fails, the RM affinity still exists. The user or node can access another IMS immediately if the Logon exit routine (DFSLGNX0) or the Sign-on exit routine (DFSSGNX0) allows it, but resource status is not recovered and local status is deleted on the failed IMS at restart.

LOCAL is the default if a resource structure or shared queues are not used.

RM is not required. If RM is not active, RM affinity is not enforced.

NONE Status Recovery Mode

No status is saved in RM or local log records. At logon or signon, significant status is cold. STSN, conversation, and fast path status is automatically nonrecoverable.

RM and a resource structure are not required.

Recoverability of specific resource types

One way to specify the recoverability of specific resource statuses (conversation status, STSN status, and Fast Path status) is to use the parameters in the DFSDCxxx PROCLIB member.

The following parameters apply to the recoverability of conversation status, STSN status, and Fast Path status.

RCVYCONV= YES | NO
Specifies whether conversation status is recovered. This parameter does not affect output messages. If conversation status is not recovered, the output is still recovered and delivered asynchronously. YES is the default when SRMDEF=GLOBAL or LOCAL. NO is the default and YES is invalid when SRMDEF=NONE.

RCVYFP= YES | NO
Specifies whether Fast Path status and Fast Path output are recovered. YES is the default when SRMDEF=GLOBAL or LOCAL. NO is the default and YES is invalid when SRMDEF=NONE.

You must specify YES if you specify that STSN status is recovered.

RCVYRESP= YES | NO
Specifies whether full-function response mode is recovered after either a terminal logoff or a user signoff. RCVYRESP=NO is the default. RCVYRESP= YES is invalid when SRMDEF=GLOBAL or NONE or when a CSL Resource Manager (RM) is active.

RCVYSTSN= YES | NO

Specifies whether STSN status is recovered for STSN terminals (SLU P, Finance, and ISC). This parameter affects only the recovery of STSN sequence numbers and does not affect output messages. YES is the default when SRMDEF=GLOBAL or LOCAL. NO is the default and YES is invalid when SRMDEF=NONE.

Related reference

"Fast Path recovery" on page 399
Recovering Fast Path transactions depends on the status recovery mode, the Fast Path recovery, the STSN recovery, and where the Fast Path transaction runs.

Fast Path recovery

Recovering Fast Path transactions depends on the status recovery mode, the Fast Path recovery, the STSN recovery, and where the Fast Path transaction runs.

The following table lists the recovery and status of Fast Path transactions based on these criteria. If the transaction runs locally without going through the EMH queue, IMS cannot recover Fast Path status globally in RM. In the situation where the status recovery mode is GLOBAL but the transaction is running locally, the status recovery mode is temporarily changed to LOCAL with the terminal or user having RM affinity.

Table 73. Determining Fast Path recoverability

Status recovery mode	Fast Path recovery (RCVYFP)	STSN recovery (RCVYSTSN)	Local Fast Path (DBFHAGU0)	Shared EMH queues (DBFHAGU0)
LOCAL or GLOBAL	NO	NO	Status and messages discarded	Status and messages discarded
LOCAL or GLOBAL	NO	YES	INVALID	INVALID
LOCAL	YES	NO	Status and output recovered locally. STSN cold started.	Status and output recovered locally. STSN cold started.
GLOBAL	YES	NO	Status and output recovered locally. STSN cold started.	Status and output recovered globally. STSN cold started.
LOCAL	YES	YES	Status and output recovered locally. STSN recoverable.	Status and output recovered locally. STSN recoverable.
GLOBAL	YES	YES	Status and output recovered locally. STSN recoverable.	Status and output recovered globally. STSN recoverable.

Recovery modes for XRF terminals

Status recovery modes affect each Extended Recovery Facility (XRF) terminal class differently during a takeover in the XRF environment.

Operating in an XRF environment does not affect status recovery modes. However, recovery of class-1 terminals is slightly different when using status recovery modes in a sysplex environment compared to a an environment not using RM.

Class-1 terminals

IMS relies solely on local log records for an XRF takeover, even if the status recovery mode is GLOBAL. When session takeover is complete, IMS reconciles any differences in status between RM and the new active system. The GLOBAL mode provides the benefit of a user or terminal being able to log on to another IMS in the IMSplex following normal termination, but for class-1 terminals, LOCAL mode may be more appropriate.

If the status recovery mode is LOCAL, RCVYRESP=YES, and RM is not used, full-function response mode is recovered for class-1 terminals.

Class-2 terminals

IMS uses local log records during a takeover if status recovery mode is LOCAL and RM if status recovery mode is GLOBAL. The new active system logs active terminals on, but user signon is required. IMS uses local log records to initiate logon, but determines recovery based on the status recovery mode of the user or terminal.

Class-3 terminals

IMS does not automatically take over these terminals. Recovery is based on the status recovery mode.

Related concepts

“Planning for the Extended Recovery Facility (XRF)” on page 405
An *XRF* complex enables you to rapidly resume end-user services when a failure occurs.

Status recovery of TM resources

IMS saves recoverable statuses for LTERMs, nodes, users, and user IDs. This topic lists the statuses saved for each.

LTERM recovery status

IMS saves the following LTERM recoverable statuses:

- LTERM name
- EDIT=UC (upper case translation specification)
- Node or user owner

IMS saves the following LTERM command significant statuses:

- `/ASSIGN SAVE`
- `STOP`

Node recovery status

IMS saves the following recoverable statuses:

- Node name
- Device type
- Allocated LTERM name
- Allocated user name

IMS saves the following command significant statuses:

- `EXCLUSIVE`
- `MFSTEST`
- `STOP`
- `TRACE`

IMS saves the following end-user significant statuses:

- Conversation
- Fast Path
- STSN
- Full-function response mode after a terminal logoff or a user signoff, but only if RM is not used, SRMDEF=LOCAL, and RCVYRESP=YES. Full-function response mode is not recovered after a termination or failure of IMS.

User recovery status

IMS saves the following user recoverable statuses:

- User name
- User ID
- Allocated LTERM names
- Allocated node names
- Autologon parameters

IMS saves the following user command significant statuses:

- `EXCLUSIVE`
- `MFSTEST`
- `STOP`
- `/CHANGE AUTOLOGON SAVE`

IMS saves the following user end-user significant statuses:

- Conversation
- Fast Path
- Full-function response mode after a terminal logoff or a user signoff, but only if RM is not used, SRMDEF=LOCAL, and RCVYRESP=YES. Full-function response mode is not recovered after a termination or failure of IMS.

User ID recoverable status

IMS saves the following LTERM recoverable statuses:

- User ID
- Terminal Name

Planning for security

To prevent unauthorized use of a terminal in the IMS network, you can use RACF (or an equivalent product).

RACF is a licensed program available under the z/OS operating system.

RACF allows you to control access to:

- Physical terminals
- Logical terminals
- Transactions
- Commands

If you do not use RACF security, IMS allows only certain commands to be entered at user terminals (excluding the master terminal). This is called *default terminal security*.

Using RACF, you can design security profiles based on user ID and you can define two levels of security for your network:

- You can control the use of the terminals connected to your network.
- You can control the resources that can be accessed from the terminal.

You control use of a terminal by signon verification security. For example, a terminal user enters an identifier as a parameter on a /SIGN command or in response to a DFS3649 message. You can use RACF, an exit routine, or both to validate the signon. The user ID is logged with each input and output message and with each database change.

Related concepts

Potential use of IMS commands (Operations and Automation)

Related reference

Terminal security defaults for IMS type-1 commands (Commands)

Authorizing transactions in a TM network

During transaction authorization, RACF checks to ensure that the terminal user is authorized to enter the transaction.

Related concepts

IMS security (System Administration)

Using RACF to secure transactions

To control which users can issue which transactions, define the controlled transactions to RACF as TIMS class, and grant authorization to RACF-defined users or groups of users.

You can also use System Monitoring Facility (SMF) logging to track the successes and failures of transaction authorization. Using RACF, request auditing capabilities for your transaction security profiles.

Using the Transaction Authorization exit routine (DFSCTRN0)

You can use the Transaction Authorization exit routine (DFSCTRN0) to check the validity of a transaction that is entered by a user who signed on using the /SIGN command.

You can use the Transaction Authorization exit routine in conjunction with RACF.

Related reading: For more information on using the Transaction Authorization exit routine (DFSCTRN0), see *IMS Version 15 Exit Routines*.

Authorizing commands in a TM network

For command authorization, you can use RACF and the Command Authorization exit routine (DFSCCMD0).

Related concepts

IMS security (System Administration)

Using RACF to secure commands

To control which users can issue controlled commands, define controlled commands to RACF as CIMS class, and grant authorization to RACF-defined users or groups of users.

You can use SMF logging to track the successes and failures of command authorization. Using RACF, request auditing capabilities for your command security profiles.

Using the command authorization exit routine (DFSCCMD0)

You can use the Command Authorization exit routine (DFSCCMD0) to restrict keywords and parameters by editing the input command buffer.

The Command Authorization exit routine checks the validity of commands. You can use this exit routine with RACF.

Related reference

Command Authorization exit routine (DFSCCMD0) (Exit Routines)

Transaction command security

Transaction command security can provide another level of access control.

Transaction command security is indirectly related to a terminal. If the terminal user enters a transaction to start a program that issues IMS commands, both the transaction code and the set of commands that program can issue must be authorized.

Password security

Password security requires that a transaction or a command that is entered at a terminal have a password. This allows you to implement secondary security to verify that the user who is issuing a specific transaction or command is authorized to do so.

Using RACF and a security profile based on user IDs

RACF provides the REVERIFY option, which requires your password to be entered when a command or a transaction is entered. You need to use RACF for signon authorization and include the REVERIFY parameter as APPLDATA when defining the transaction or command to RACF. You must specify RVFY=Y as an execution parameter in order to use REVERIFY.

Security for APPC/IMS

APPC/IMS requires security using the System Authorization Facility (SAF) interface to RACF, or an equivalent security environment. RACF is optional for remote transactions from LU 6.2 application programs.

APPC/IMS supports both the Transaction Authorization exit routine (DFSCTRN0) and the Command Authorization exit routine (DFSCCMD0).

Restrictions:

- APPC/IMS does not support the `/SIGN` command, because it is not required in order to validate the user ID. z/OS validates user IDs when using RACF; therefore, each APPC/IMS message has a validated user ID.
- For IMS commands entered from remote LU 6.2 application programs: if you do not use RACF or the Command Authorization exit routine (DFSCCMD0), the default command security allows only the following four commands:
  - `/BROADCAST`
  - `/LOG`
  - `/RDISPLAY`
  - `/RMLIST`

  To allow other commands, use DFSCCMD0 or RACF.

Related reference

DL/I calls for transaction management (Application Programming APIs)

Security for ETO

You can use RACF security for command and transaction authorization on both static and ETO terminals.

Related concepts

"Planning a high-security environment with ETO" on page 78

ETO enhances the security of your IMS system. You can customize the ETO security features for your installation needs.

Planning for Fast Path terminals

To have a Fast Path-capable system, specify Fast Path support on the FPCTRL macro.

IMS systems with a very high transaction rate use Fast Path's expedited message handler (EMH) facility. EMH is a performance option that expedites message processing by imposing restrictions on message lengths and segmentation. LU 6.2 terminals can use EMH; no definitions or specifications are required.

With Fast Path, an EMH buffer is acquired from the EMH buffer pool when the first eligible input message is received for a Fast Path transaction. The buffer remains allocated to the terminal for future use until the session terminates or the user signs off. If an EMH buffer is allocated to the terminal, it is reused if it meets the requirements of the next input call. If the message requires a larger buffer, Fast Path swaps it for a larger one.

If, at entry to IMS, an input message satisfies all of the following criteria, it is edited in a Fast Path EMH buffer instead of in a full-function message queue buffer.

- Terminal is not FES capable
- Terminal is not the MTO
- Input is single segment
- Input is non-MFS edited
- Input is not an IMS command
- Execution of at least one Fast Path transaction is scheduled using Fast Path

Exception: If the message is not a Fast Path transaction, the message is moved to a full-function message queue buffer.

When a message with all of these characteristics is received, an EMH buffer is allocated to the terminal. The EMH buffer remains allocated to the terminal until either the user signs off or the session terminates. The buffer is used repeatedly for Fast Path transactions until a larger EMH buffer is required for special applications. When a larger buffer is needed, the current EMH buffer is swapped for the larger-size EMH buffer. The smaller EMH buffer is released and returned to the EMH pool.

The EMHL parameter in the IMS control region initialization (also known as the startup parameter) specifies the default EMH buffer size for all Fast Path transactions for all Fast Path-eligible terminals. A Fast Path transaction can specify an EMH buffer that is larger than the default. Specify larger application EMH buffer sizes on the APPLCTN or TRANSACT stage_1 macros by using the FPATH=*size* parameter when you generate the system. If the EMH buffer pool is exhausted, message DFS3971 is sent to the input terminal.

Buffers are extracted from the EMH buffer pool, which expands and contracts dynamically. The size of the pool depends on the number of terminals that are concurrently entering Fast Path transactions and the buffer sizes that are required to satisfy each request.

You can say FPATH=No or Yes on the APPLCTN and TRANSACT macros, or you can specify FPATH=*size*, where *size* is the EMH buffer size that is required to run the transaction. FPATH=*size* implies FPATH=Yes. The minimum EMH buffer size is 12 bytes, and the maximum is 30720 bytes.

MSDBs are available to ETO terminals, unless they have terminal-related keys.

The OPTIONS keyword on the TERMINAL macro or ETO descriptor is used to:

- Declare FORCRESP or TRANRESP so that Fast Path-eligible terminals operate in response mode

- Specify PAGDEL for automatic page deletion where appropriate

You can use non-VTAM terminals to enter Fast Path transactions. For non-VTAM terminals, the LINE macro must indicate response mode by using the RESP=TERM parameter.

Related reference

Terminals and equipment supported by IMS 15 (Release Planning)

Planning for the Extended Recovery Facility (XRF)

An *XRF* complex enables you to rapidly resume end-user services when a failure occurs.

You can also use XRF to switch all transaction and database processing from one IMS system to another for scheduled maintenance or during off-peak periods.

When you do not use XRF, a high level of availability with short interruptions in end-user service can be difficult to achieve when IMS fails. Termination processing must complete, and the cause of the failure determined before restart can take place. A restart can be time-consuming if IMS regions must be restarted and terminal sessions reestablished.

An XRF configuration consists of a primary system, called the *active IMS*, and a secondary system, called the *alternate IMS*. The alternate IMS tracks the activities of the active IMS in the same or in a different central processing complex (CPC). The active IMS processes the IMS workload in the same way that it would without XRF. It does not do any additional processing or logging. Using the log, the active IMS informs the alternate IMS of its activities. Surveillance mechanisms alert the alternate IMS to problems in the active IMS. The active IMS is not aware of the activities of the alternate IMS.

The alternate IMS does not process any transactions; it maintains the status of the active IMS, and monitors it for any indications of trouble. The alternate IMS tracks resource changes in the active IMS, and updates its control blocks and buffers while continuously changing its status of readiness. In this manner, work can be quickly shifted from the active IMS to the alternate IMS without interruption. This shift of work from the active IMS to the alternate IMS is known as the takeover process. The following figure shows an example of a simple XRF complex.

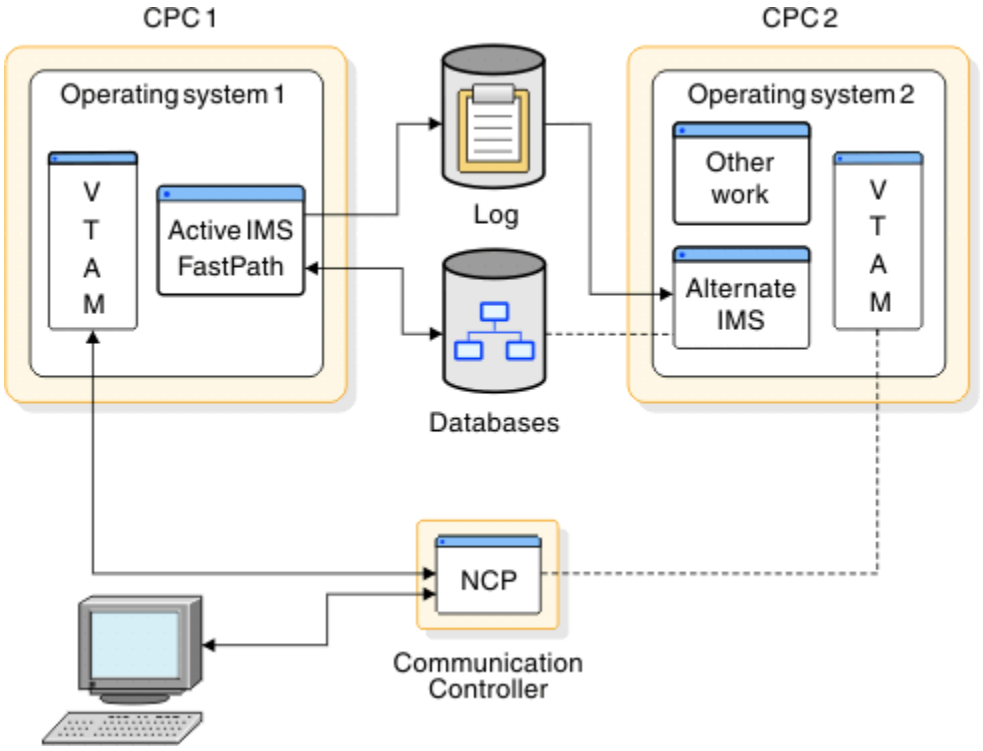


Figure 50. Sample XRF complex

Related concepts

Extended Recovery Facility Overview (System Administration)

Using MNPS or USERVAR in XRF complexes

You can use IMS multinode persistent sessions (MNPS) support for XRF. Due to performance limitations, this is not a recommended solution.

You can implement one of two types of XRF complexes:

- An XRF complex that uses VTAM MNPS with an MNPS ACB.
- An XRF complex that uses a user variable (USERVAR) with USERVAR tables.

Both types of XRF complex provide the same increased availability for IMS, but differ in how they manage terminal sessions and the takeover process. Other differences include:

- XRF complexes that use a USERVAR have additional hardware requirements.
- VTAM and the network terminals use different names to communicate with IMS in each type of complex:
  - In an XRF complex that uses an MNPS, IMS is known to VTAM and the network terminals by the name of the MNPS ACB (the MNPS name).
  - In an XRF complex that uses a USERVAR, IMS is known to VTAM by the name of the APPLID ACB (the APPLID name) and to the network terminals by the USERVAR.

XRF complexes that use MNPS still have an APPLID ACB, but it is not used for the purposes of XRF.

Related concepts

Extended Recovery Facility Overview (System Administration)

Terminals in an XRF complex

XRF supports all terminals that your IMS users are currently using. XRF provides emergency support with a minimum of effort from you.

XRF requires that you specify keywords on several system definition macros or ETO descriptors. Unless you want to change the switching priority of a terminal from the defaults, or to change the terminal type sessions, you need not change any terminal definition macros.

XRF automatically gives the terminals at your installation the best service that the terminal's characteristics allow. This topic describes these characteristics.

XRF provides support to the following three types of terminals:

Class–1
: At takeover, the sessions persist. The sessions are switched transparently to the alternate IMS without losing the session.

Class–2
: At takeover, the sessions end. The alternate IMS automatically tries to reestablish the sessions. The users are briefly without service.

Class–3
: At takeover, the sessions end. The operator or the end users can establish new sessions with the new active IMS if the terminals are physically connected to it.

Because APPC/IMS devices are Class 3, LU 6.2 conversations that are in progress at the time of failure are not preserved during a takeover. A new LU 6.2 conversation must be allocated from the new active IMS after a takeover occurs. The new active IMS processes APPC/IMS activity exactly the same as the old active IMS did prior to takeover. The new active IMS has the same LU name as the old active IMS.

Class-1 terminals

To take full advantage of XRF, define most of the terminals in your complex as Class-1 terminals. Class-1 terminals are those terminals defined on the UNITYPE keyword of the TYPE macro as either:

- SLUTYPE1
- SLUTYPE2
- SLUTYPEP
- FINANCE

Class-1 terminals are handled differently depending on whether you use VTAM Multinode Persistent Session (MNPS) support.

Class-1 terminal recovery in an XRF system with MNPS

When you use XRF with MNPS, class-1 terminals do not have backup sessions on the alternate IMS. Instead, when a class-1 session is initiated in the active IMS, IMS tells VTAM to track both the session state and data traffic information.

During an XRF takeover, IMS requests that VTAM verify the class-1 terminal sessions and uses the data traffic information to transparently restore sessions. Higher priority terminals are restored before lower priority terminals.

The terminal types listed in the table in “Specifying terminal support” on page 408 are eligible for class-1 service if they are owned by an XRF-capable VTAM that is MNPS enabled. To enable VTAM for MNPS, specify PERSIST=MULTI on the APPL definition.

Class-1 terminal recovery in an XRF system with USERVAR

In an XRF system with USERVAR, the alternate IMS maintains backup sessions for all class-1 terminals with open sessions on the active IMS. These sessions are established during the tracking phase when a session on the active IMS is initiated. In the event of a takeover, the alternate IMS switches all class-1 terminals to the backup sessions.

From the terminal user's point of view, only one session with IMS exists. From the NCP point of view, two host sessions exist. The actual messages are only passed between the node and the active IMS. The alternate IMS tracks the status of the message activity by monitoring the IMS log.

When the active IMS establishes or terminates a terminal session with class-1 terminals, the alternate IMS establishes or terminates its backup session with the terminal. When the active IMS fails, the alternate IMS takes over the terminal session (without losing control from the viewpoint of the node) by changing the mode of the established backup session from BACKUP to ACTIVE. When the session is switched, the NCP sends the new active IMS its view of the terminal's status. IMS compares this with its own status record to decide what, if any, recovery action to take. When the REVERIFY operand is used with RACF, the user must sign on again.

The terminal types listed in the table in “Specifying terminal support” on page 408 are eligible for class-1 service if they are both:

- Owned by an XRF-capable VTAM
- Connected to an XRF-capable NCP in a 37x5 Communication Controller

Class-2 terminals

Class-2 terminals do not have backup-session support on the alternate IMS, but they are restarted automatically by IMS after takeover.

When a Class-2 terminal session is established on the active IMS, the alternate IMS tracks the session initiation and termination by using log records. When the active IMS terminates abnormally, the alternate IMS tries to establish a new session with the network resources that were active before the failure.

When IMS tries to reestablish a session with a terminal that has no available network path, the attempt fails. Before IMS can reestablish a session on a locally attached terminal, the operator must switch the

terminals to the new active IMS. If the complex has many of these kinds of terminals, you might consider giving the operator control before the takeover actually proceeds. At this time, the operator could perform the switch, allowing a faster recovery for sessions on these terminals. To give the operator this control, use the AUTO parameter in member DFSHSBxx in IMS.PROCLIB.

See the table in "Specifying terminal support" on page 408 for terminals that qualify as class-2.

If you have an application terminal that communicates with a programmable controller in an XRF complex, you might need to add code to the communications portion in order to achieve the highest level of transparency available. Additionally, for the ISC terminal, you must define a session for the link that connects the ISC terminal to an XRF IMS.

Class-3 terminals

Class-3 terminals do not have backup session support on the alternate IMS. Their terminal sessions must be restarted manually on the new active IMS after takeover, either by the MTO or by user logon.

All IMS terminals are eligible for class-3 service. Class-1 and class-2 terminals that have indicated no switching should be done are eligible for class-3. This is done by specifying BACKUP=NO on the system definition macro or ETO logon descriptor.

Class -3 terminals use the existing procedures to connect with the new active IMS after takeover. If no path to the new IMS systems exists, reestablishing the sessions might not be possible. All LU 6.2 conversations on the failing IMS must be reallocated after a takeover.

Related concepts

Terminals in an XRF complex (System Administration)

Related reference

"Recovery modes for XRF terminals" on page 399
Status recovery modes affect each Extended Recovery Facility (XRF) terminal class differently during a takeover in the XRF environment.

"Coding ETO descriptors" on page 80
IMS uses ETO descriptors to dynamically build terminal structures and user structures.

Macros used in IMS environments (System Definition)

Specifying terminal support

The following table summarizes the keywords you can use to change the priority or terminal type of various devices.

All terminals that IMS supports can be used in an XRF complex. The terminal characteristics determine the type of support a terminal can have in an XRF complex, with class-1 having the best support and class-3 having no support.

Table 74. Specifying terminal support

Device	Class	BACKUP keyword on TERMINAL macro or ETO logon descriptor
SLUTYPE1 (3286) SLUTYPE2 (3278,3279,...) SLUTYPEP (4700,8100, SERIES 1,...) FINANCE 3601	Eligible for class 1,2,3	For class-1 service, BACKUP=(1-7,YES). For class-2 service, BACKUP=(1-7,NO). For class-3 service, BACKUP=NO.

Table 74. Specifying terminal support (continued)

Device	Class	BACKUP keyword on TERMINAL macro or ETO logon descriptor
LUTYPE6 (ISC) MSC (VTAM) NTO (leased line) 3270 (VTAM) SPOOL line groups Locally attached devices	Eligible for class 2,3	For class-2 service, BACKUP=(1-7,NO). For class-3 service, BACKUP=NO.
LU 6.2	Eligible for class 3	Not Applicable

The primary and the secondary master terminals are special types of terminals that exist simultaneously on both the active and alternate IMS systems. They are not treated as class-1 or class-2 terminals. The master terminals on the alternate IMS are not disrupted by a takeover. After takeover, the master terminal on the alternate IMS processes the undelivered messages from the active IMS's master terminal.

The BACKUP keyword on several macros in the system definition statement or ETO logon descriptor specifies the actual support a terminal is to receive. Your installation determines the switching priority of the terminals, with level 1 being the lowest priority and level 7 the highest. Although IMS orders VTAM requests by priority, the requests might be completed in any order because of unpredictable factors in the network, such as pacing, line speeds, and congestion.

If BACKUP=(1-7,YES), the terminal is automatically taken over without losing session control. If the terminal is connected to an XRF-capable VTAM, this constitutes class-1 support.

For both class-1 and class-2 terminals, the default switching priority is 4.

If BACKUP=NO, this terminal must be restarted manually after takeover. This forces class-3 support.

If options on the BACKUP parameter are set incorrectly, IMS resets the incorrect option at session initialization. For example, if you specify BACKUP=(1-7,YES) for an ISC terminal, IMS will reset YES to NO because the highest class of support that an ISC terminal can receive is class-2.

XRF with USERVAR and the BACKUP parameter

For XRF systems with USERVAR, BACKUP=(1-7,YES) also tells the alternate IMS to maintain backup sessions for all class-1 terminals. The maximum number of terminal backup sessions that can be established on the alternate IMS is specified in the NCP BUILD statement with the BACKUP=n option. If the (n+1) class-1 terminal user tries to log onto the active IMS, the logon is denied.

If BACKUP=(1-7,NO), this terminal is automatically restarted after takeover, but no BACKUP session is established. This forces class-2 support.

Planning for Rapid Network Reconnect (RNR)

Rapid Network Reconnect (RNR) automatically reconnects IMS VTAM terminal sessions across outages (IMS, z/OS, CPC, or VTAM) and subsequent IMS restarts on the same or another CPC within an IMSplex. The use of RNR can provide greater availability of VTAM sessions and eliminate the need to clean up sessions and restart IMS after an outage.

Note: The following topics apply only to RNR systems and should not be confused with XRF systems, which also use VTAM multinode persistent sessions (MNPS).

Specifying levels of support

RNR support can be specified at three levels: IMS execution parameters; or ETO logon descriptor OPTIONS= parameter; or by session using the Logon exit routine (DFSLGNX0).

RNR support is specified in the DFSDCxxx PROCLIB member by indicating RNR=ARNR, RNR=NRNR, or RNR=NONE. ARNR activates RNR and specifies that a basic session reconnect will be performed automatically by IMS during the /START DC process following an IMS or VTAM restart. NRNR specifies that automatic session reconnect will be activated, but not used unless overridden by the logon descriptor or the logon user exit routine. Unless overridden, the session will be terminated following restart of IMS or VTAM. NONE will not activate RNR at all.

If RNR is specified without a parameter (RNR= without ARNR or NRNR), NRNR is the default. If RNR is not specified at all (no RNR=), RNR will not be activated just as if RNR=NONE was specified.

RNR support can also be indicated on the ETO logon descriptor using the OPTIONS= ARNR|NRNR parameter. The RNR support specified at this level overrides that set at the system level using IMS execution parameters. If IMS was initialized with RNR unspecified, then specification of RNR support on the ETO logon descriptor is ignored.

RNR support can be indicated on the Logon exit routine (DFSLGNX0) using the LGOPT=LGOARNR parameter. The RNR support specified at this level overrides that set using the ETO logon descriptor or the system level execution parameters. If IMS was initialized with RNR unspecified, then specification of RNR support on DFSLGNX0 is ignored.

Using the DFSDCxxx PROCLIB member, you can also specify the maximum time for session persistence following an IMS or VTAM failure. The parameter for specifying this time is PSTIMER. PSTIMER can be set from 1 to 86400 seconds. The default setting is 3600 (1 hour). If 0 is indicated, then no timer is used and session persistence will continue indefinitely.

Note: If the VTAM START option for MNPS, HPRPST, is set to a lower value than PSTIMER, then VTAM START overrides the PSTIMER setting.

Changing levels of support

RNR support can be turned on or off between sessions or between IMS restarts, and it can be changed at any level: in the IMS PROCLIB member, in the ETO logon descriptors, and in the Logon exit.

After RNR has been set on or off, it is not applied until the next session cold start. For IMS terminals (non-ISC, Finance and SLUP), the result is that the last RNR option selected becomes active at each logon. For ISC, Finance, and SLUP terminals, the result is that the last RNR option selected spans multiple logons and IMS restarts and becomes active only at the next session cold start.

Persistent Session Tracking

VTAM Persistent Session Tracking is provided for both single-node persistent sessions (SNPS), and multinode persistent sessions (MNPS).

The level of VTAM persistent session support desired for IMS is specified on the APPL definition statement using the PERSIST=MULTI|SINGLE parameter.

VTAM SNPSs have the following characteristics:

- Reconnect must be on the same CPC as the original IMS.
- Only IMS failures and reconnects are supported.

VTAM MNPSs have the following characteristics:

- Reconnect may be on another CPC in an IMSplex.
- IMS, VTAM, z/OS, and CPC failures and reconnects are supported.

If you use VTAM MNPS, keep in mind that VTAM end nodes must be running in an APPN/HPR (High Performance Routing) network environment. In addition, all VTAM instances must be connected to a Parallel Sysplex coupling facility using the ISTMNPS structure.

APPC Persistent session support is provided by APPC/MVS. However, APPC conversations are not automatically restarted following an outage.

Termination of Persistent Session Tracking

If a session terminates before IMS has closed the VTAM access control block (ACB), VTAM persistent session tracking, and the ability of IMS to reconnect, are terminated on all terminals.

An IMS-initiated shutdown could be executed, for example, by issuing the `/CHECKPOINT FREEZE` or `/STOP DC` or `/CLSDST` commands.

A VTAM session is also disconnected if VTAM persistent session tracking is prematurely terminated due to a non-IMS-initiated session close if that session close occurred before an IMS restart completion, which can happen for one of the following reasons:

- Remote operator-initiated session termination; executing, for example, the `/RCL` command
- Network operator-initiated session termination; executing, for example, the `VARY NET` or `INACT` commands
- Timeout of persistent session tracking indicated by the PSTIMER value

If IMS is defined for RNR on MNPS, the VTAM Network Operator can determine whether persistence is currently active by using the `VTAM DISPLAY ID` command.

When shutting down VTAM, the `HALT NET` or `HALT NET, QUICK` commands terminate session persistence, and the `HALT NET, CANCEL` command maintains session persistence.

IMS shutdown and RNR

When IMS is shut down using the **`MVS MODIFY`** command for IMS, RNR quickly terminates the IMS network, and VTAM persistence is retained.

After the **`/START DC`** command is issued, IMS then automatically reconnects all sessions specified with ARNR that were still active at ABEND; sessions that were specified with NRNR are terminated.

When IMS is shut down using the **`/STOP DC`** command followed by the **`/CHECKPOINT`** command, VTAM terminals are disconnected for each session, resulting in no automatic session reconnect following an IMS restart.

Using RNR with XRF or VGR

RNR is not supported on a system configured with XRF. Attempting to activate RNR on an XRF-configured system produces a warning message indicating that execution will continue with RNR disabled. RNR can be used in conjunction with VGR; however, IMS support for RNR takes precedence over VGR support.

If a **`/START DC`** command is entered during an IMS restart, and RNR=NRNR, then the session is terminated and VGR, if active, performs the appropriate level of affinity and terminal status management. If a **`/START DC`** command is entered during an IMS restart, and RNR=ARNR, then the session is scheduled for reconnection and a new session with another IMS cannot be established by invoking VGR. If IMS is cold-started, then all active VTAM sessions are terminated and all active VGR affinities and statuses are deleted.

Restriction: RNR does not support OTMA connections.

Terminal reconnect protocols

The reconnect protocol used for terminals with RNR activated depends on the kind of work in progress and the types of terminals in use at the time of the outage.

The following table shows the reconnect protocol used for each type of terminal following a session outage. All terminals are assumed to have RNR activated. If a session cannot be reconnected, error message DFS2050 or DFS2055 is sent to the Master Terminal Operator.

Table 75. Rapid Network Reconnect protocols

Terminal type	Connect protocol	Terminal subtype	Message generated
SLU1	SNA CLEAR SNA SDT	• Static sessions • ETO printers	DFS3650[1]
		ETO non-printers	DFS3649
SLU2	SNA CLEAR	Static sessions	DFS3650
	SNA SDT	Static sessions	DFS3649
NTO	SNA CLEAR	Static sessions	DFS3650
	SNA SDT	ETO sessions	DFS3649
SLU0 (Non-SNA 3284 and 3286)	SNA UNBIND SNA BIND	Static sessions	DFS3650
		ETO sessions	DFS3649
SLU0 (Finance and SLUP)[2]	SNA CLEAR SNA STSN SNA SDT	ETO sessions	No message
ISC (Primary Half-Session)	SNA UNBIND SNA BIND SNA STSN SNA SDT		No message
ISC	SNA UNBIND		No message
ETO w/ ALOT=0[3]	UNBIND UNBIND		No message

Notes:

1. Message DFS3649 indicates that signon is required. Message DFS3650 indicates that no signon is required.
2. Reconnect is available only for Single-Node Persistent Sessions.
3. Signon data must be supplied as logon user data or by the DFSLGNX0 exit.

Signon security

Depending on whether RNR is activated or not and the types of terminals you are using, signons or logons of VTAM sessions may be required following IMS restarts.

If RNR is not activated, then both logon and signon of sessions are required. If RNR is activated, then the following types of terminals require signon after a session reconnect:

- SLU0 (non–printer, 3270, non-SNA)
- SLU1 (non-printer only)
- SLU2
- NTO

If RNR is activated, then the following types of terminals sign on automatically after a session reconnect:

- SLU0 (Finance/3600)

- SLU0 (SLUP)
- SLU1 (3284, 3286, non-SNA)
- ISC (LU 6.1)

Chapter 24. Defining the network

Performing a system definition for an IMS terminal network includes several tasks.

The tasks include:

- Defining the names of the terminals
- Defining the terminal device types
- Specifying optional parameters, such as buffer sizes

The IMS network consists of static and dynamic terminals (ETO and LU 6.2 devices). You do not need to define your ETO or LU 6.2 devices to IMS.

Although only static terminals in the IMS network are defined by system definition macros, both static and dynamic terminals must be defined to VTAM. You must coordinate several aspects of the IMS requirements with parameters that are specified during VTAM network generation. The following topics highlight those coordination activities, but do not attempt to relate the separate IMS system definition in a detailed way with the series of VTAM and NCP definitions.

Related concepts

Defining terminals with data communication macros (System Definition)

Preparing for the operational network

In establishing the network, you must incorporate the IMS system definition requirements into the corresponding VTAM generations. You should also track the overall progress and the hardware installation schedules.

The principal activities involved in establishing the IMS network consist of:

- Gathering the IMS requirements for physical terminals.

  In addition, gather the following information regarding terminal requirements:

  - Organize input data for stage–1 IMS system definition for static terminals.
  - Define ETO descriptors and display screen characteristics for dynamically allocated terminals.
  - Define LU 6.2 descriptors.
  - Use the terminal profiles as part of the documentation for the IMS system design.

- Matching the terminals that are specified for IMS with those available for wider use by the installation.

  Multiple users can exist on high-function terminals, and the terminals might not be dedicated to the IMS online system. You need to be able to identify these terminals to the network generation personnel and to identify expected usage by the end users.
- Matching IMS system definition parameters to their counterparts in network generation.

  Correct device function might require the matching of IMS system definition parameters with VTAM and NCP network parameters. Be especially careful when defining buffer sizes. IMS and network buffer sizes must be compatible. Although system definition in IMS for ETO and LU 6.2 devices are not necessary, you must define them to VTAM.
- Monitoring the hardware installation plan.

  Create a plan for tracking the progress of hardware installation and network generation to ensure that the change to production mode is not jeopardized. This tracking might involve interacting with hardware specialists who install and maintain terminal hardware.
- Anticipating terminal installations.

  You might want to predefine terminals to IMS in anticipation of their installation.

© Copyright IBM Corp. 1974, 2020

Related tasks

"Administering the Extended Terminal Option" on page 69
The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

"CPI Communications and APPC/IMS" on page 23
These topics introduce CPI-Communications and APPC/IMS. The topics discuss how CPI-Communications driven application programs function and how to administer APPC/IMS and use APPC/IMS with the CPI Communications interface to build CPI application programs.

Coordinating IMS definition and network definition

Provide input to VTAM and NCP regarding the network requirements described in the following topics.

Using IMS as a host subsystem

You must define the IMS control region to VTAM as an application. Your installation needs to decide on a name. This name is used in the definition of application nodes that are added to the SYS1.VTAMLST system data set.

Example: The following is an example of a control region definition:

```
IMS  APPL AUTH=(ACQ),PRTCT=password
```

This example shows that a password is required in order to start communication with the subsystem. Use the PASSWD keyword on the COMM macro to specify, for IMS system definition, the corresponding password. If the VTAM definition does not use IMS as the name for the IMS subsystem, code the chosen subsystem name in the COMM macro for IMS, using the APPLID keyword. The default APPLID name is IMS.

Defining VTAM nodes

The local node names are also added into SYS1.VTAMLST.

You need to take special care that the exact terminal names used in the IMS system definition are repeated in the local list members of that data set. Other characteristics of the terminal are specified in the VTAM entries, and you must ensure that no inconsistencies exist between IMS and VTAM parameters. Associated with the node name are:

- The use of program function keys or selector pen
- Keyboard characteristics
- Screen sizes
- Model number
- LU type
- Transmission service level

IMS verifies screen size, model, LU type, and transmission service for dynamic terminals. If these are not correct, a session is not established.

Recommendation: If you are a DC administrator, assure that the VTAM definitions are correct. These definitions are used to select ETO descriptors and MFS format characteristics. IMS system definition sets these definitions for static terminals and ignores VTAM definitions.

Related concepts

Defining VTAM terminals (System Definition)

Estimating VTAM storage requirements

Examine the full set of terminal options that are planned for the IMS network and select those communication options that might have an effect on the VTAM storage requirements.

For example, choose the appropriate:

- Protocol (BID or NOBID)
- Type of response patterns
- Number of active nodes

Using your terminal profiles, you can respond to the need for system programming information to estimate storage requirements for IMS support.

Determining VTAM buffer pool values

Provide the VTAM system programmer with IMS input and output message sizes. The values you select for the number and size of receive-any buffers, as specified on the RECANY keyword of the COMM macro, are of special importance.

You must specify the largest number of receive-any buffers that are required in order to support the VTAM network. You can override the size with the execution parameter, RECASZ, and the number with the execution parameter, RECANY. Session initiation fails for terminals that are added to the system if they require a larger receive-any buffer size than the size that is currently specified. You must restart the IMS system, specifying the increased buffer size, before the terminal can establish a session with IMS.

Determining the NCP buffer pool values

The NCP buffer pool sizes and thresholds are based on the VTAM buffer pool information.

Work with the system programmer to assess the volume of predicted traffic for application programs. No specific IMS requirements exist for NCP system definition parameters.

Determining static and dynamic terminal signon requirements

All VTAM-defined terminals can sign on by providing user data with the session initiation request, regardless of whether signon is required for the terminal.

You cannot sign on a user to more than one terminal simultaneously, unless you specify SGN=G, M, or Z on the startup procedure to allow multiple signons. Otherwise, IMS issues message DFS3649A or DFS2467I.

A user cannot enter signon data from an output-only device through logon or a **/SIGN** command. When signon data is omitted at session initiation for an output-only device that requires a signon, message DFS2085I is sent to the MTO.

Multiple signons

You cannot use the same name for dynamic LTERMs as for static LTERMs that are defined during system definition.

You can allow users to sign on concurrently to one or more terminals, static or dynamic, by specifying SGN=M in either the IMS or DCC procedures.

The IMS master terminal receives a security violation message, DFS286, for rejected signon attempts. If you do not want to receive the message, set SECCNT to 0 on the COMM macro, the IMSGEN macro, or the SECCNT initialization EXEC parameter.

Session status message—DFS3650I

When signon is achieved or if signon is not required, message DFS3650I is sent to the user indicating the status of the session with IMS.

Exception: The following terminals do not receive these messages:

Autologon terminals
SLU-1 terminals running in unattended mode
ISC
SLU P
3600/Finance

In addition, if NOTERM is specified on the TERMINAL macro or on the ETO user descriptor, no DFS3650I message is received.

The information presented on the DFS3650I message panel shows whether the user is in conversation mode. This status panel also shows whether user output security exists for the terminal.

Related tasks

"Extended Terminal Option (ETO)" on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

Related reference

COMM macro (System Definition)

IMSGEN macro (System Definition)

Related information

DFS3650I (Messages and Codes)

Checking requirements for LOGON MODE tables

When a static VTAM terminal is connected to IMS and a session is initiated, VTAM needs to know the LOGON MODE identifier.

A VTAM LOGON MODE table contains the SNA bind parameters for the session and the identifier (the LOGMODE parameter value, which is used at logon) for that terminal. Also, a default identifier for each SDLC-connected terminal exists.

These considerations apply to static VTAM terminals.

Coordinating LOGON MODE identifiers

When you are deciding on the LOGON MODE identifiers, you need to:

- Match the naming conventions that are being used for the VTAM network in your installation. Your choice of a name might need to follow this convention.
- Arrange for automatic starting of IMS-remote terminals, or have them controlled by the VTAM operator. Otherwise, choose names that are meaningful to a remote terminal operator, if in order to initiate a session the remote terminal operator (RTO) enters the VTAM LOGON command.
- Choose names that are helpful to the master terminal operator if sessions are to be started with the /OPNDST command. If a terminal can be operated with different SNA protocols, the name should help distinguish between the operating modes.
- Use the MODETBL keyword on the TERMINAL macro or ETO logon descriptor to specify the default identifier that is to be used in the following situations:
  - The default identifier is not used.
  - The terminal operator does not specify the identifier in the LOGON command.
  - The master terminal operator does not specify the identifier in the /OPNDST command.

Examples

The code below illustrates the nature of the LOGON MODE table entries that are described to VTAM for a 3270 SDLC connection or a 4730 device. IMS defines these as SLU 2 and SLU P respectively.

```
SE3270   MODETAB
IBMS3270 MODEENT LOGMODE=S3270,FMPROF=X'02',TSPROF=X'02',       X
           PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000'
MT4730   MODEENT LOGMODE=MT4730,FMPROF=X'04',TSPROF=X'04',      X
```

```
            PRIPROT=X'B1',SECPROT=X'B1',COMPROT=X'6080',         X
            RUSIZES=X'0000'
          MODEEND
```

The operator can enter either the VTAM LOGON command or use the VTAM USSTAB command for installation-determined logon syntax.

The following example specifies a KEYWORD=*value* format:

```
U3270 USSTAB
      USSCMD     CMD=IMS,REP=LOGON
      USSPARM    PARM=APPLID,DEF=IMS
      USSEND
      END
```

Logon requirements for the master terminal

Because IMS automatically logs on the master terminal after the IMS START command is issued, you must code the MODETBL keyword on the LU statement for the physical device that is used as the MTO if the VTAM default is not to be used.

When the MTO is using the /OPNDST command, a MODE operand applies to all node names presented in the command.

Related tasks

“Extended Terminal Option (ETO)” on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

Specifying initial VTAM configurations

Based on your IMS requirements, start lists (ATCSTRyy) and configuration members (ATCCONxx) are placed in the SYS1.VTAMLST library. This information should reflect those terminals that are to be activated at VTAM startup, and those VTAM nodes that are known to VTAM at startup.

For ISC (LU 6) devices and MSC VTAM links, if the session is started from another subsystem, the session is not automatically recovered. Session failure messages are sent to the MTO indicating that you must manually restart the session.

You should emphasize flexibility in the IMS requirements, and try to reduce the requirements for a large part of the network to be immediately available. This might entail more elaborate startup instructions for start lists and configuration members. This might also be reflected in your MTO instructions and the extent to which the **/OPNDST** command is used. The delayed start up of network sections can overlap with IMS processing in order to allow IMS to start up more quickly. Also, the network startup sequence can be executed by an automated operator program executing in IMS.

Using SON/COS support in IMS

Session outage notification (SON) and class of service (COS) are facilities of VTAM and SNA that allow IMS to recognize a session outage.

IMS attempts to restart the session for recoverable session failures. Both SON and COS must be specified in VTAM to be available for IMS use. In addition, the ASR option must be specified for the IMS node using the system definition process or the /CHANGE command.

The IMS SON and COS support are available for all the VTAM SNA terminals and MSC VTAM links. When multiple virtual routes exist between IMS and the other LU, use an alternate route to avoid a network outage.

The SON facility enables VTAM to inform IMS that a session failure has occurred. The SON facility gives installations the option of allowing IMS to automatically restart sessions (without end-user intervention) if a recoverable session failure occurs in the network.

You need to code SONSCIP=YES on the VTAM APPL definition statement for the IMS application in order to activate automatic session restart after a session outage. If you decide not to include the SONSCIP definition for IMS, VTAM and IMS proceed with session termination without automatic session restart.

The COS facility allows VTAM to control selection of actual routes by designating a set of virtual communication routes based on speed of traffic, data type, and security considerations. These sets of routes are defined in the VTAM COS table. The mode table entry that is used for session establishment can specify a COS entry name.

When the set of session BIND parameters associated with the terminal contains a COS name, these specified virtual routes are scanned to determine which one can be used for the session. If the selected virtual route fails, restart of the session causes the COS set to be scanned again for use of a different virtual route for session reestablishment.

For program LUs (as well as Finance, LU P, MSC, and ISC), in-flight messages can be recovered and retransmitted as necessary, thereby preserving the integrity of the message exchange across the restart.

For device LUs, in-flight messages might be lost or duplicated.

SON and COS support preserves the session setup options (BIND parameters), as well as the class of service associated with the failing session by using the same mode table entry name in order to reestablish a new session.

Restrictions: Because the boundary node NCP cannot distinguish an upstream route failure from a failure of a host CPC, SON and COS support cannot be used for terminals in XRF configurations that have backup sessions.

Related reference

/CHANGE commands (Commands)

Starting an IMS network

To make IMS ready to receive VTAM logon (session initialization) requests, use the IMS **/START** command with the DC keyword.

Before a session with IMS can be established, VTAM and NCP must be active.

The **/START DC** command tells VTAM to pass to IMS any queued VTAM logon requests to IMS, as well as logon requests for any logical unit that is defined to VTAM as belonging to IMS.

The **/START DC** command activates the following processes:

- Initiates IMS data communication processing
- Opens the VTAM access method control block
- Enables the IMS VTAM Logon exit routine

Any logon requests VTAM receives before the IMS **/START DC** command but after the IMS VTAM access method control block (ACB) has been opened are queued in VTAM until the **/START DC** command is completed. If VTAM is active when IMS is initialized, the IMS VTAM ACB is opened. If VACBOPN=DELAY has been specified, then the VTAM ACB open is delayed until the **/START DC** command is entered.

Use the **/START APPC** command to start APPC/IMS.

Session initiation

A *session* is the logical connection between a logical unit (such as a terminal) and a VTAM application program (such as IMS). A session must be established before data can be transmitted between a logical unit and IMS.

Session initiation is requested in one of five ways:

- The terminal operator enters the LOGON sequence. VTAM verifies the command and passes the request to IMS.

  The terminal operator can identify IMS in the logon sequence with the following names:

  - The APPLID name, if the terminal operator is requesting a session with a specific IMS.
  - The MNPS ACB name, if the terminal operator is requesting a session with an XRF with MNPS system.
  - The USERVAR, if the terminal operator is requesting a session with an XRF with USERVAR system.
  - The generic resource name, if the terminal operator is requesting a session with a generic resource group.
- The z/OS VTAM network operator requests session initiation on behalf of the logical unit by using the VTAM **VARY** command with the LOGON option. VTAM processes the request and passes it to IMS.
- VTAM passes a logon request to IMS for each logical unit that is defined to VTAM as belonging to IMS.
- The IMS master terminal operator requests session initiation for a logical unit by entering the IMS **/OPNDST** command.
- ETO autologon initiates the session and supplies the appropriate user data, based on user descriptors and exit routines.

For LU 6.2, session initiation occurs automatically as conversations are allocated. The IMS MTO can request delivery of queued LU 6.2 output by issuing the **/ALLOCATE** command. When using LU 6.2, remember that the IMS application name (which matches the LU name) for LU 6.2 conversations is different than the LU name IMS has for non-LU 6.2 types, and that the **/START DC** and **/START APPC** commands work independently.

Regardless of how session initiation is requested, identical processing occurs when IMS receives the request.

Related concepts

"Sharing printers between systems" on page 441
If you plan to share the use of a printer other than the master, IMS allows an online IMS system and another subsystem (possibly another IMS system) to alternate their use of the printer.

"Sharing printers using ETO" on page 97
Several users can share printers by using the same output terminal.

IMS transaction types and transaction states

Transactions are the most common type of data that is sent from a logical unit to IMS.

IMS supports two kinds of transactions—update and inquiry.

Definitions:

- An *update* transaction can modify a database.
- An *inquiry* transaction can look at data in a database, but it cannot change or update it.

You define transactions as update or inquiry during IMS system definition.

An additional attribute is defined for inquiry transactions—recoverable or irrecoverable.

Definitions:

- *Recoverable-inquiry transactions* are always recoverable, regardless of which element in the network fails.
- *Irrecoverable-inquiry transactions* are not recovered after an I/O error condition occurs or at IMS system restart.

All update transactions are recoverable. All Fast Path transactions must be defined as recoverable, but they can be either inquiry or update.

An LU 6.2 transaction is recoverable only if the asynchronous protocol is used to initiate the transaction. IMS conversational transactions from LU 6.2 programs are not recoverable in the local system, but they are recoverable across the MSC link.

IMS treats an irrecoverable transaction in the same manner as a recoverable transaction, except that all processing that required to achieve recoverability is eliminated. As a result, irrecoverable transactions

require less processing time, but they could be lost in the event of a network failure (for example, line failure, CPC failure, or queue failure).

Transactions created for queuing only by the Destination Creation exit routine (DFSINSX0) have a status of DYN. The only purpose of a queue-only transaction is to queue a message to the shared queues. Queue-only transactions are not recovered at restart, unless they are stopped, or were not yet checkpointed.

Determining transaction states

A transaction can be in any one of a number of states.

The possible transaction states are:

LOCK
A transaction in LOCK state does not receive messages. Scheduled messages containing this transaction code are stopped.

Restriction: The `/LOCK TRANSACTION` command cannot be used with Fast Path-exclusive transactions, but it can be used with Fast Path-potential transactions.

MODSTOPPED
A transaction in MODSTOPPED state cannot receive input, because online change processing is in progress. A `/MODIFY COMMIT` command sets this status. A CPI Communications driven transaction cannot be marked MODSTOPPED.

PSTOP
A transaction in PSTOP state cannot be scheduled; however, the transaction continues to be processed until the limit count is reached. If the limit count is large, the processing interval is long. To ascertain the status of the transaction, use the `/DISPLAY` command; to alter the status of the transaction, use the `/ASSIGN` command.

PURGE
A transaction in a PURGE state has had its input messages stopped.

QSTOP
A transaction in QSTOP state cannot be entered during the time between the completion of a `/MODIFY PREPARE` command and the completion of the corresponding `/MODIFY COMMIT` or `/MODIFY ABORT` command. Transactions that are known to be affected by the content of an online change are rejected; that is, the transactions are to be changed or deleted, or they could access databases or programs that are to be changed or deleted. By using the `/DISPLAY MODIFY` command, you can cause a list of transactions that are affected by a current online change to be displayed. At the terminal, such a transaction is rejected with message DFS3470. If the transaction uses a Fast Path routing code that is changed or deleted, the rejection message is DFS3471. A CPI Communications driven transaction is never in a QSTOP state.

STOP
A transaction in STOP state is stopped. The queuing and scheduling of messages that are destined for a transaction or class of transactions are stopped. However, output can still be queued if it originates from an application program.

USTOPPED
USTOPPED is a status value that is set when an application program attempts to use an IMS DL/I database resource that is not available, and the program terminates abnormally with abend U3033. The USTOPPED condition is not set for a CPI Communications driven program. The CPI Communications driven application program is abnormally terminated with an abend U0125.

Defining VTAM for Rapid Network Reconnect (RNR)

To use Rapid Network Reconnect (RNR) for VTAM sessions, you must define VTAM for the level of persistent session support desired using the APPL statement for IMS, and assess user security requirements and exposures and define an appropriate level of RNR support for terminals.

Related concepts

"Planning for Rapid Network Reconnect (RNR)" on page 409

Rapid Network Reconnect (RNR) automatically reconnects IMS VTAM terminal sessions across outages (IMS, z/OS, CPC, or VTAM) and subsequent IMS restarts on the same or another CPC within an IMSplex. The use of RNR can provide greater availability of VTAM sessions and eliminate the need to clean up sessions and restart IMS after an outage.

Defining the level of persistent support

VTAM Persistent Session Tracking is provided for both single-node persistent sessions (SNPS), and multinode persistent sessions (MNPS). You must indicate which level of persistent support is needed for RNR.

MNPS allows VTAM sessions to be reconnected to another CPC in a sysplex, if necessary. SNPS requires that VTAM sessions be reconnected to the same CPC they were connected to when the outage occurred.

The level of VTAM persistent session support desired for IMS is specified on the APPL definition statement using the PERSIST=MULTI|SINGLE parameter. The default setting is PERSIST=SINGLE. If no PERSIST specification is entered, PERSIST=SINGLE is assumed.

Defining the level of RNR support

RNR support can be defined at three levels: the system level, the terminal level, and the session level.

To use all three levels of control over RNR support, you must:

- Update the IMS execution parameters to activate RNR.
- Update the appropriate IMS ETO Logon descriptors for dynamic terminal support.
- Update the DFSLGNX0 Logon exit support for dynamic override of the RNR option on a session by session basis.

To activate RNR at the system (and default) level, you must specify in the DFSDCxxx IMS.PROCLIB member whether to activate RNR by indicating either RNR=ARNR (Activate RNR) or RNR=NRNR (No RNR).

Using the PSTIMER parameter in the DFSDCxxx IMS.PROCLIB member, you can also specify the maximum time for session persistence following an IMS or VTAM failure. PSTIMER can be set from 1 to 86400 seconds. The default setting is 3600 (1 hour). If 0 is indicated, then no timer is used and session persistence continues indefinitely.

Note: If the VTAM START option for MNPS, HPRPST, is set to a lower value than PSTIMER, it overrides the PSTIMER setting.

To control RNR support for ETO dynamic terminals, use the OPTIONS= ARNR|NRNR parameter in the ETO logon descriptor. The RNR support specified at this level overrides that set at the system level using IMS execution parameters.

To control RNR support on a session by session basis, use the LGOPT=LGOARNR|LGONRNR parameter for the Logon exit routine (DFSLGNX0). The RNR support specified at this level overrides that set using the ETO logon descriptor or the system level execution parameters.

The parameters of DFSLGNX0 are documented in its source code.

Chapter 25. Editing and formatting IMS messages

IMS uses two methods to edit and format messages to and from terminals: Message Format Service (MFS) and basic edit routines.

IMS provides samples of user-written exit routines that can be designed to edit:

- Input and output from a terminal
- Transaction codes
- Input message fields
- Input message segments
- Message switching

This topic presents an overview of the advantages of MFS, introduces MFS control blocks for message formatting, and summarizes the characteristics of the different devices MFS supports and the responsibilities of the MFS administrator.

Restriction: MFS does not support LU 6.2 devices or OTMA. The LU 6.2 Edit exit routine (DFSLUEE0) is provided for both input and output messages from LU 6.2 devices when the implicit API support is used. The OTMA Input/Output Edit user exit (OTMAIOED) is provided for both input and output messages from OTMA.

Related reference

OTMA Input/Output Edit user exit (DFSYIOE0 and other OTMAIOED type exits) (Exit Routines)

LU 6.2 Edit exit routine (DFSLUEE0) (Exit Routines)

Message Format Service

Message Format Service (MFS) is an IMS facility that formats messages to and from terminals, so that IMS application programs need not deal with device-specific characteristics in input or output messages.

MFS formats messages to and from user-written programs in remote controllers and subsystems, so that host application programs need not deal with terminal-specific characteristics of the remote controller.

MFS uses control blocks that the user specifies to indicate to IMS how input and output messages are arranged.

- For input messages, MFS control blocks define how the message that is sent by the device to the application program is arranged in the program's I/O area.
- For output messages, MFS control blocks define how the message that is sent by the application program to the device is arranged on the screen or at the printer. Data, such as literals that appear on the screen but not in the program's I/O area, can also be defined.

In IMS systems, data that is passed between the application program and terminals or remote programs can be edited by MFS or basic edit. The facilities provided by MFS depend on the type of terminals or secondary logical units (SLUs) your network uses.

MFS allows application programmers to deal with logical messages instead of device-dependent data; this simplifies application development. The same application program can deal with different device types using a single set of logic, whereas device input and output are varied for a specific device type. The presentation of data on the device or operator input can be changed without changing the application program. Full paging capability is provided by IMS for display devices. Input messages are created from multiple screens of data.

A program using MFS need not be designed for the physical characteristics of the device that is used for input and output messages unless it uses very specific device features. Even when these features are used, the program can request that MFS assist in their presentation to the program or the device.

© Copyright IBM Corp. 1974, 2020

MFS supports SLU-type devices SLU-1, SLU-2, SLU-P, Finance, and LU 6.1. MFS also supports older devices, including IBM 3270 and 3600.

For IBM 3270 or SLU-2 devices, device control characters or orders can be sent directly from or received by the program using the MFS bypass function. This gives the application program more direct control of the data stream. The program uses reserved format names that cause MFS to bypass the edit of:

- the output message
- the next input message that is received from the display terminal

Both logical- and physical-paging facilities are provided for the IBM 3270 and 3604 display stations; these facilities allow the application program to write large quantities of data that MFS can divide into multiple display screens on the terminal. The terminal operator has the capability to page forward and backward to different screens within the message.

MFS components

MFS has several components.

The MFS components include:

MFS Language utility
Message editor
MFS pool manager
MFS Service utility
MFSTEST pool manager
Message Format Service Device Characteristics Table (MFSDCT) utility (DFSUTB00)

The MFS Language utility is executed offline to generate control blocks and place them in a format control block data set named IMS.FORMAT. The control blocks describe the message formatting that is to take place during message input or output operations. They are generated according to a set of utility control statements.

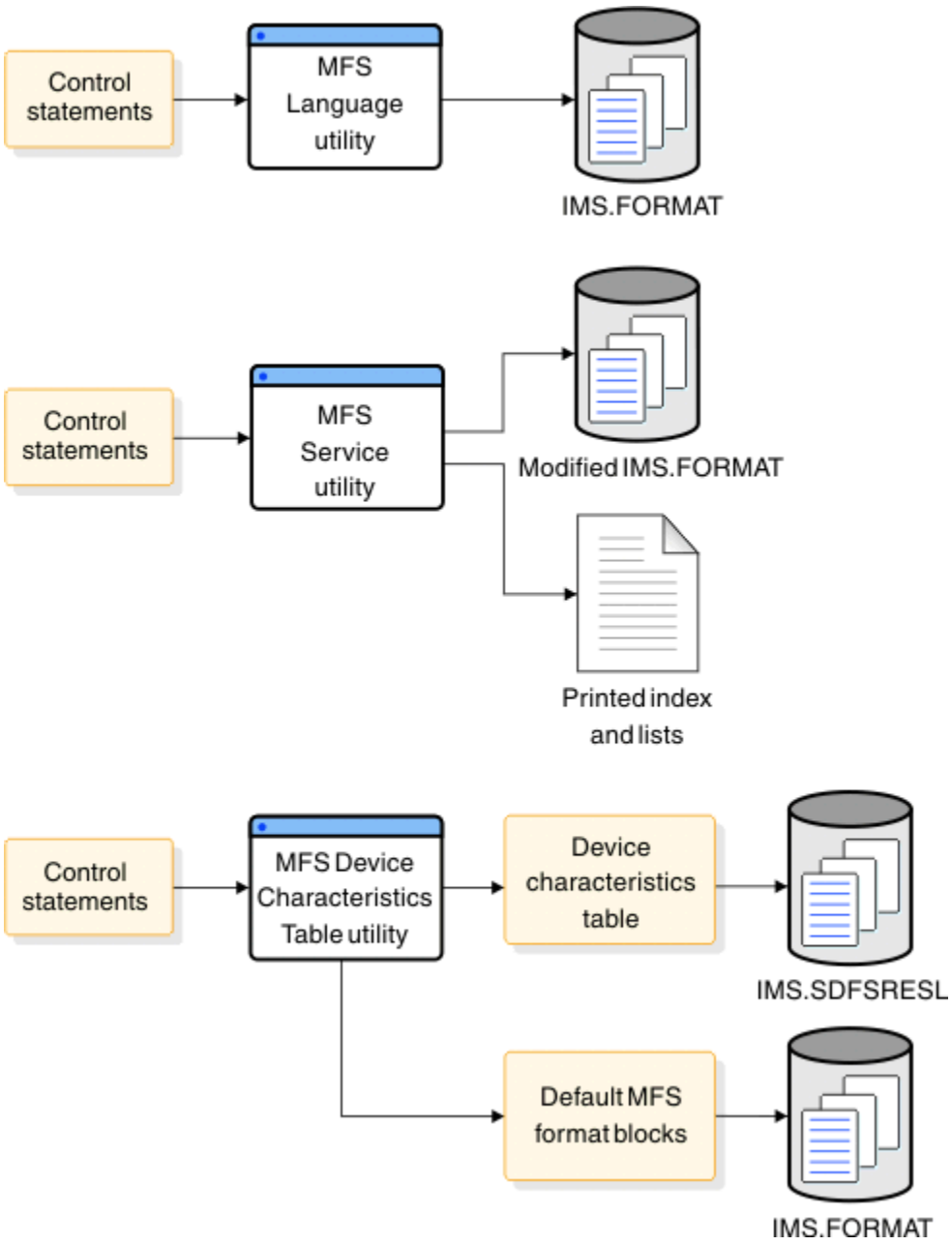


Figure 51. MFS utilities and their output

Note: In MFS test mode, the MFS Language utility can run at same time as the online IMS control region. However, you must use the online change procedure to modify MFS formats when not in IMS test mode.

The following figure shows the MFS online environment. The steps listed following the figure correspond to the numbers in the figure.

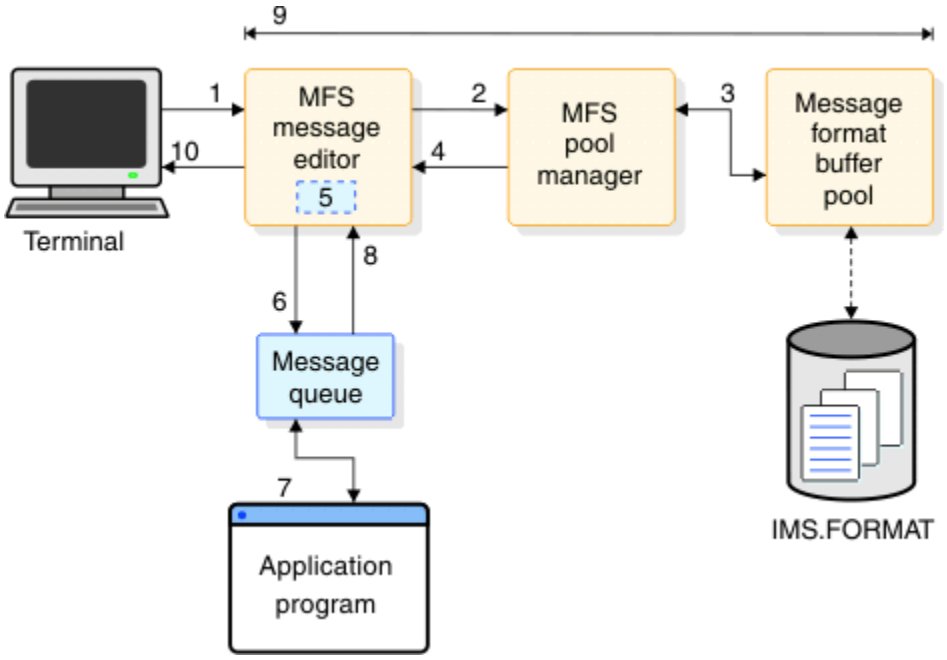


Figure 52. Overview of the MFS online environment

1. Input message is sent to the MFS message editor from the terminal.
2. MFS message editor requests pointer to MFS blocks from the MFS pool manager.
3. MFS pool manager checks the message format buffer pool to see if the blocks exist in the pool. If the blocks do not exist, the MFS pool manager reads the blocks from IMS.FORMAT into the buffer pool.
4. MFS pool manager sends the address of the MFS blocks to the MFS message editor.
5. MFS message editor formats the input message for the application program.
6. MFS message editor sends the formatted input message to the message queue to be processed.
7. Application program processes the message and sends the output message to the message queue.
8. Output message is sent from the message queue to the MFS message editor.
9. MFS processes the output message for the terminal just as it processed the input message (steps "2" on page 428 through "6" on page 428).
10. The formatted output message is sent to the terminal.

The following figure show the MFS test environment. The steps listed following the figure correspond the numbers in the figure.

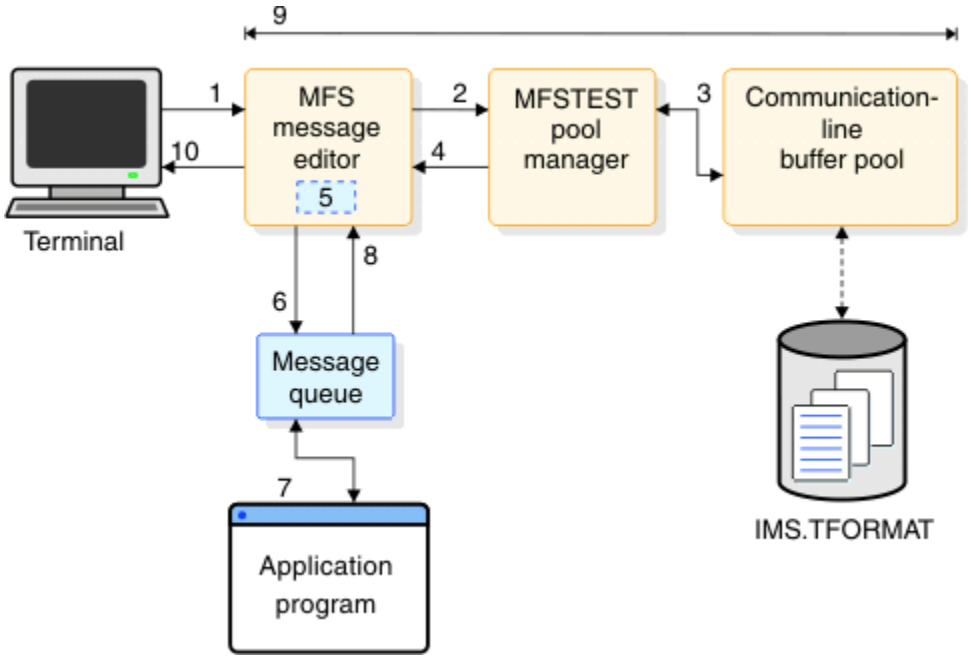


Figure 53. Overview of the MFS test environment

Note: You must use the **/TEST MFS** commands to begin MFS test mode.

1. Input message is sent to the MFS message editor from the terminal.
2. MFS message editor requests pointer to MFS blocks from the MFSTEST pool manager.
3. MFSTEST pool manager checks the communication-line buffer pool to see if the blocks exist in the pool. If the blocks do not exist, the MFS pool manager reads the blocks from IMS.TFORMAT into the buffer pool.
4. MFSTEST pool manager sends the address of the MFS blocks to the MFS message editor.
5. MFS message editor formats the input message for the application program.
6. MFS message editor sends the formatted input message to the message queue to be processed.
7. Application program processes the message and sends the output message to the message queue.
8. Output message is sent from the message queue to the MFS message editor.
9. MFS processes the output message for the terminal just as it processed the input message (steps "2" on page 429 through "6" on page 429).
10. The formatted output message is sent to the terminal.

The message editor and MFS pool manager operate online during the normal production mode of operation. The message editor performs the actual message formatting operations using the control block specifications.

Two other MFS components, an MFS Service utility and an MFSTEST pool manager, are available to support optional MFS operations.

The MFS Service utility provides a method for additional control of the format control block data sets. It executes offline and is able to create and maintain an index of control blocks for online use by the MFS pool manager.

The MFSTEST pool manager replaces the MFS pool manager in order to support the optional MFSTEST mode of operation. The **/TEST** command with the MFS keyword places a logical terminal into MFSTEST mode. For each terminal in MFSTEST mode, combining temporary format blocks with the use of other blocks that are already in production mode allows new applications and modifications to existing applications to be tested without disrupting production activity.

Related reference

MFS Service utility (DFSUTSA0) (System Utilities)

MFS Language utility (DFSUPAA0) (System Utilities)

MFS Device Characteristics Table utility (DFSUTB00) (System Utilities)

Administering MFS

To take full advantage of the flexible message formatting options offered by MFS and to ensure efficient MFS operation, an MFS administrator should be appointed.

The MFS administrator should be responsible for MFS implementation and administration and should coordinate MFS application design and programming for the installation.

The responsibilities of an MFS administrator include:

- Establishing procedures for the submission of MFS control block requests by application development personnel.
- Establishing procedures and controls for the application of changes to the IMS.TFORMAT library.
- Defining MFS control blocks most efficiently in keeping with the requirements of the specific application and the overall system.
- Minimizing device character transmission, sharing MFS control blocks, and ensuring the most efficient use of MFS without jeopardizing application requirements or operator considerations.
- Establishing and documenting operator guidelines and system standards for MFS. The many options that MFS offers can result in confusing practices, unless you establish and follow standard procedures. Be sure to standardize certain aspects of format design in order to minimize terminal operator training and error rates.
- Deciding if and how the optional index directory should be used and determining buffer pool requirements.
- Monitoring the use of the MFS control blocks and of the MFS buffer pool with the IMS **/DISPLAY** command and IMS Monitor report output, and modifying MFS parameters as required.
- Making end users aware of the operating characteristics of the different device types and terminal subsystems.
- Informing others about the differences between the various partition formats.
- Establishing and informing others about naming conventions and guidelines. In particular, the MFS administrator should be able to discuss naming conventions for the partition descriptor blocks and the sizes of the display screen, the viewports, and the display characters.
- Communicating information on conventions for and restrictions on MFS formats.
- Defining screen sizes and feature combinations that are not included in the IMS stage–1 system definition.
- Creating the MFS device characteristics table control statements for processing by the MFSDCT utility (DFSUTB00). The MFS device characteristics table entries and default format control blocks are used for ETO terminals.
- Defining input message field edit routines and segment edit routines. MFS and all MFS-supported devices are able to use message edit routines. You can use these exit routines for such common editing functions as numeric validation or conversion of blanks to zeros.

  IMS provides a sample of both a field edit and a segment edit routine.
- Determining whether MFS verifies the protected fields that are returned by 3270 and SLU2 devices. If MFS detects that the content of the protected fields that are returned from the devices are different from the content of the fields that were transmitted to the devices, MFS ignores the returned fields. The MFS administrator specifies whether MFS verifies the protected data fields by configuring the DFSDCxxx member of the IMS PROCLIB data set.

The MFS administrator should be technically competent in all aspects of IMS relative to MFS:

- Online transaction processing

- IMS API for message processing
- Operation with remote controllers
- MFS implementation, device characteristics, and capabilities
- Interpretation of MFS statistics and related IMS Monitor report output

The administrator should also be familiar with the hardware and remote programs for SLU-P, Finance remote programs, or ISC subsystems if such programs are going to operate with MFS by using distributed presentation management.

In addition, because one administrative responsibility is minimizing device character transmission, the administrator should be familiar with the terminal hardware characteristics.

An MFS administrator must communicate with IMS system administrators and application developers, as well as programmable workstation developers and end users. The administrator must be able to enforce installation standards and to modify application specifications for MFS control blocks when necessary to benefit overall system performance. The procedures of related programming groups should recognize this authority of the MFS administrator.

Related concepts

Input message field and segment edit routines (Application Programming APIs)

Advantages to using MFS

Two primary advantages to using MFS are that it simplifies the development and maintenance of terminal-oriented application systems and improves online performance.

To simplify IMS application development and maintenance, MFS performs many common application program functions and gives application programs a high degree of independence from specific devices or remote programs.

With the device independence offered by MFS, one application program can process data to and from multiple device types while still taking advantage of their different capabilities. Thus, MFS can eliminate or minimize the changes in application programs when new terminal types are added.

MFS makes it possible for an application program to communicate with different types of terminals without having to change the way it reads and builds messages. When the application program receives a message from a terminal, how the message appears in the program's I/O area is independent of the kind of terminal that sent it; the appearance depends on the MFS options specified for that program. If the next message that the application program receives is from a different type of terminal, the user does not need to do anything to the application program. MFS shields the application program from the physical device that is sending the message in the same way that a database program communication block (PCB) shields a program from the data in the database and how it is stored.

Other common functions MFS performs include left or right justification of data, padding, exit routines for validity checking, time and date stamping, page and message numbering, and data sequencing and segmenting. When MFS performs these functions, the application program handles only the actual processing of the message data.

The following figure shows how MFS can make an application program device-independent by formatting input data from the device or remote program for presentation to IMS, and by formatting the application program data for presentation to the output device or remote program.

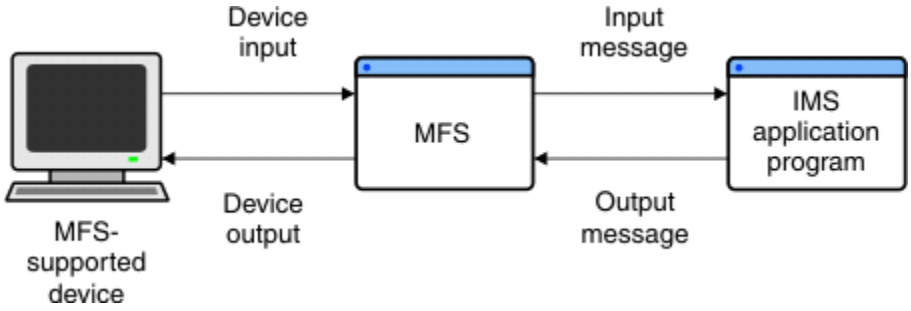


Figure 54. Message formatting using MFS

MFS also improves online performance of a terminal-oriented IMS by using control blocks that are designed for online processing. The MFS control blocks are compiled offline, when IMS is not being executed, from source language definitions. MFS can check their validity and make many decisions offline to reduce online processing. In addition, during online processing, MFS uses look-aside buffering of the MFS control blocks in order to reduce CPU usage and the channel costs of input/output activity.

Because MFS control blocks are reentrant and can be used for multiple applications, online storage requirements are reduced. Optional main-storage indexing and anticipatory fetching of the control blocks can also reduce response time. IMS gains additional performance improvements, because multiple I/O operations can execute concurrently in loading the format blocks from the MFS format library.

In addition, MFS uses z/OS paging services to reduce page faults by the IMS control region task.

Finally, MFS can reduce use of communication lines. Compressing and transmitting only the required data reduces line load and improves both response time and device performance.

Related reference

Transaction Manager exit routines (Exit Routines)

MFS control blocks

Users can specify four types of MFS control blocks to format input and output for the application program and the terminal or remote program.

Definitions:

- *Message Output Descriptors (MODs)* define the layout of messages that MFS receives from the application program.
- *Device Output Formats (DOFs)* describe how MFS formats messages for each of the devices with which the program communicates.
- *Device Input Formats (DIFs)* describe the formats of messages MFS receives from each of the devices with which the program communicates.
- *Message Input Descriptors (MIDs)* describe how MFS formats messages so that the application program can process them.
- *Message descriptors* are both MIDs and MODs.
- *Device formats* are both DIFs and DOFs.

Because each MOD, DOF, DIF, and MID deals with a specific message, both a MOD and DOF must exist for each unique message a program sends, and both a DIF and MID must exist for each unique message a program receives.

Overview of MFS components and operation

MFS has the following components:

- The MFS language utility, which generates control blocks from user-written control statements and places them in a library called IMS.FORMAT
- The MFS service utility, which is used for maintenance of the control blocks in IMS.FORMAT

- The MFS message editor, which formats messages according to the control block specifications generated by the language utility
- The MFS pool manager, which keeps the MFS control blocks that are required by the message editor in the main-storage MFS buffer pool
- The MFSTEST Pool Manager, which replaces the MFS pool manager when the language utility is being used in test mode

IMS online change also plays an important part in updating the MFS libraries, even though it is not part of MFS. Briefly, online change allows the control block libraries to be modified while the IMS control region is executing.

Related concepts

The online change function (System Administration)

Creating MFS formats with SDF II

SDF II is an interactive tool for designing and generating MFS formats.

SDF II does not replace MFS, but it does make developing and maintaining MFS formats easier. Because SDF II uses a panel editor for designing and testing formats, it frees the MFS programmer from some of the tasks associated with coding MFS source statements.

With SDF II, application programmers and analysts who might not know the special requirements of MFS can perform a part of the programming job that would otherwise call for specialized knowledge. SDF II uses a panel editor, such as the one shown in the following screen shot, to define and test a panel.

```
                              DEFINE FORMAT

Format . . . . . .Positions 1-75 or 80, Lines 1-24 of 24
Marks: V - C . L , S +                              Contents: FORMAT
001
002                 ************************************
003                 ** E M P L O Y E E   P A Y R O L L **
004                 ************************************
005
006  LAST NAME:                      FIRST NAME:
007
008  EMPL NO:
009
010  SOC SEC NO:
011
012  RATE OF PAY:
013
014
015
016  INPUT:
017
018
019
020
021
022
023
024
PF1=HELP     2=SPLIT     3=END       4=RETURN      5=RFIND      6=CHANGE
PF7=UP       8=DOWN      9=SWAP     10=LEFT       11=RIGHT     12=CURSOR
```

After testing the panel, the SDF II user can automatically generate the MFS source code, shown in the example code below, that is needed for the format definition. Using MFS only, the programmer would need to code these statements manually.

```
DOF
PAYF     FMT
         DEV       TYPE=(3270,2),FEAT=IGNORE,DSCA=X'00A0'
         DIV       TYPE=INOUT
         DPAGE     CURSOR=((5,15))
         DFLD      '**********************',POS=(1,21)
         DFLD      '*  EMPLOYEE PAYROLL  *',POS=(2,21)
         DFLD      '**********************',POS=(3,21)
         DFLD      'LAST NAME:',POS=(5,2)
```

```
LNAME    DFLD      POS=(5,15),LTH=16
         DFLD      'FIRST NAME:',POS=(5,36)
FNAME    DFLD      POS=(5,48),LTH=16
         DFLD      'EMPL. NO:',POS=(7,2)
EMPNO    DFLD      POS=(7,11),LTH=6
         DFLD      'SOC SEC NO:',POS=(9,2)
SSN      DFLD      POS=(9,14),LTH=11
         DFLD      'RATE OF PAY: $',POS=(11,2)
RATE     DFLD      POS=(11,16),LTH=9
         DFLD      'INPUT:',POS=(16,2)
INPUT    DFLD      POS=(16,9),LTH=30
         FMTEND
MID
PAYIN    MSG       TYPE=INPUT,SOR=(PAYF,IGNORE)
         SEG
         MFLD      'PAYUP '     SUPPLIES TRANCODE
         MFLD      LNAME,LTH=16
         MFLD      FNAME,LTH=16
         MFLD      EMPNO,LTH=6
         MFLD      SSN,LTH=11
         MFLD      RATE,LTH=9
         MFLD      INPUT,LTH=30,JUST=R,FILL=C'0'
         MSGEND
MOD
PAYDAY   MSG       TYPE=OUTPUT,SOR=(PAYF,IGNORE)
         SEG
         MFLD      LNAME,LTH=16
         MFLD      FNAME,LTH=16
         MFLD      EMPNO,LTH=6
         MFLD      SSN,LTH=11
         MFLD      RATE,LTH=9
         MFLD      INPUT,LTH=30,JUST=R,FILL=C'0'
         MSGEND
```

SDF II is designed for use on a 3270–display device; however, SDF II can create formats for all devices supported by MFS.

Related reading: For more information on SDF II, see *SDF II General Introduction*.

Basic edit

If you do not use MFS, an IMS function called basic edit performs message editing.

For input messages, basic edit:

- Translates messages to uppercase, if specified by the EDIT=UC parameter on the system definition TRANSACT macro.
- Removes leading control characters from the first segment of each message. Leading blanks are also removed from the first segment if the message is not the continuation of a conversation or a message from a terminal in preset mode.
- Removes leading control characters from all subsequent message segments, if the message type is a transaction or a command (except the /BROADCAST command).
- Removes line control characters from all segments.
- Removes trailing carriage return and end-of-block characters from all segments of a transaction.
- Eliminates backspaces, on a one-for-one basis, from all segments when the entering or transmission of backspaces is a normal correction procedure on the entering terminal.
- Removes the password and replaces it with a blank when necessary to provide separation between the transaction code, logical terminal, or command verb, and data that follows.
- Inserts, in front of data that is entered in the first segment of a message, the transaction code or logical terminal name defined by the prior /SET command. A blank is inserted following the transaction code, if it is necessary to obtain separation between the inserted transaction code and the entered data.
- Adds a nonconversational transaction code to the first segment of the next input message, if a terminal is in conversation mode and the application terminates the conversation by inserting a nonconversational transaction code into the SPA.
- Removes the function management header (FMH), if any, that appears at the beginning of the first transmission of a chain for VTAM-supported devices.

- Deblocks message segments at each interrecord separator (IRS) control character, and discards the IRS control character for input from a SLU–1 card reader, a transmit data set (TDS), or a user data set (UDS).
- Deblocks message segments at each new line or forms-feed control character if the optional MFS editing is not selected for SLU–1 consoles. This character can be discarded, depending on the criteria previously described.
- Treats the presence of a TRN (X'35') character immediately followed by a length in the next byte as transparent data.

For output messages, basic edit:

- Changes nongraphic characters in the output message before the data goes to the device.
- Inserts any necessary idle characters after new-line, line-feed, and tab characters.
- Adds line control characters for the operation of the communication line.

For basic edit support of SLU–1 transparent data, basic edit does not alter or delete characters following the destination and password fields if transparent processing has been requested. Indicate transparent processing by specifying:

- BINPDSB1=BINTRNDS on the bind image for VTAM-type SLU–1 terminals
- Edit option BASIC=TRN on the COMPT1 parameter of the IMS TERMINAL macro or ETO descriptors for SLU–1 terminals

Related concepts

"Conversational transactions" on page 389
Conversational transaction processing allows you to retain message continuity from a given terminal, even when the program that processes the conversation is not retained in storage throughout that conversation.

IMS editing for Intersystem Communication (ISC)

IMS recognizes several options on a message-by-message basis when communicating with ISC (LU 6.1).

IMS recognizes the following options:

- Use basic edit.

  The destination process name (DPN) is BASICEDT, which is a reserved IMS name.
- Treat the data as transparent data.

  The IMS preset destination mode, which is established by the **/SET** command, is used. This option is used if a **/SET** command is in effect for a given session and a primary resource name (PRN) is not specified on the ATTACH or SCHEDULER FMH. While this option is in effect, IMS cannot recognize input commands within this input data stream.
- Do not edit data that follows the transaction code and input password fields.

  The input destination and security checking is the same as in basic edit. A / as the first character indicates an IMS command and causes the IMS preset destination mode to be bypassed or terminated. No additional editing is done on the input message following the input transaction code or LTERM name and optional password field. This is the default input edit for ISC under each of the following conditions:

  ATTACH or SCHEDULER FMH is not included in input
  ATTACH or SCHEDULER FMH does not specify a DPN
  ATTACH or SCHEDULER FMH specifies a DPN but not a PRN
- Do nothing.

  IMS uses the input PRN ATTACH parameter as the transaction code or LTERM name for the duration of the single input message. Subsequently, the original IMS preset value can be used again. Basic edit treats the input as transparent with password security already checked.

The chosen option depends on the destination process name (DPN) and the data received by IMS. The DPN is specified on the ATTACH or SCHEDULER FMH.

If a DPN of ISCEDT or its alias is specified the second through fourth options are chosen. The alias for ISCEDT is defined using the EDTNAME keyword on the COMM macro. ISCEDT is also a reserved IMS name.

Related concepts

"Editing messages" on page 459
Within IMS, the communication interface is transparent to application programs.

Transparency option

Transparent data includes programmed symbols, extended field attributes, and commands.

These unprintable characters might appear on the 3270 screen as hyphens or another representation for unprintable characters.

If IMS attempts to send transparent data to an IBM 3270 terminal that is unable to receive the data, an error is returned to IMS, and the terminal is taken out of service. To restart the terminal, the master terminal operator must issue the **/START** command.

For LU 6.1, input is processed using the transparency option of basic edit if the primary resource name (PRN) in the function management header (FMH) contains a transaction. Otherwise, the data stream is only scanned for the transaction code and optional password. The remaining data is treated as transparent data.

Related concepts

Extended field attributes for output devices (Application Programming APIs)

Unprotected screen option

When the screen is in unprotected status, IMS can send output to the terminal at any time without requiring input from the terminal. This option can be used on a terminal-by-terminal or message-by-message basis.

Bypassing MFS editing

IMS enables an IMS application program to bypass MFS formatting of an output message destined to an IBM 3270 or to SLU–2 devices by specifying reserved format names DFS.EDT or DFS.EDTN. This bypass is intended for subsystems or installation support programs that execute under IMS.

Recommendation: Do not use this bypass for IMS application programs, because the application program loses the productivity, device independence, and migration benefits associated with MFS.

When MFS is bypassed on output, the application program must construct the entire 3270 data stream, beginning with the command code and ending with the last data byte. The user might want to bypass the MFS output edit function to allow application programs to receive the 3270 input data stream from the terminal without MFS or basic editing.

Locking and unlocking the terminal keyboard

When the application uses MFS bypass and the reserved format name DFS.EDTN, IMS enables the application program to lock or unlock the terminal keyboard.

If the lock option is specified, the program is responsible for unlocking the keyboard after processing is completed. If the unlock option is specified, IMS controls the locking and unlocking of the keyboard.

Related concepts

MFS message formats (Application Programming APIs)

IMS sensitivity to nongraphic message data

The following topics describe the sensitivity IMS has to specific characters when users attempt to send and receive nongraphic data in IMS messages.

Output message segment editing

For output message segments that MFS edits, only graphic data (X'40' through X'FE') is contained in the output message that is presented to the device. Nongraphic characters, if present in the output message, are changed by MFS before the data is presented to the device.

Device control characters HT, CR, LF, NL, and BS are changed to X'00' for 3270 data streams. For all other device types, all nongraphic characters are also changed to blanks.

If the Distributed Presentation Management (DPM) option of MFS is used for SLU-P or ISC, the user can specify GRAPHIC=NO in the SEG statement. Nongraphic characters, if present in the output segment with GRAPHIC=NO specified, are presented unchanged to the remote program.

For programmable workstations that are supported through VTAM, IMS can insert function management headers (FMHs) and can perform additional editing for device control sequences when splitting a single IMS segment into multiple transmissions.

Editing of input message segments by MFS

If MFS is defined for a device, you should be aware of certain considerations.

The considerations include:

- Specify GRAPHIC=NO in the SEG statement to prevent uppercase translation on a segment if the destination requests it with the EDIT=UC specification on the system definition TRANSACT macro.
- If the first input record is from a 3600, SCS1, or SCS2 device, or from DPM-An, the segment is discarded if the final characters of the segment are:

  Two asterisks (**)
  Two asterisks followed by NL (**X'15')
  Two asterisks followed by IRS (**X'1E')
- The presence of two slashes (//) at the beginning of a message segment is considered an escape sequence.
- If the card feature is defined for an SCS1 device (with the CARD=operand in the DEV statement), all control characters are removed from magnetic card input before the data is presented to the input MFLD.

The definition of the MFS delete characters (LDEL=operand in the DEV statement) and field tab character (FTAB= operand in the DEV statement) for MFS-supported devices, excluding IBM 3270, can direct the editing of input message segments.

If the input is processed by MFS, the editing performed is dependent on the descriptions provided through the Message Format Service language utility. As input segments from the device might have no relationship to input message segments after MFS editing, the input segment from the device is not available to user-written edit routines. Input message segments after MFS editing are available to user-written edit routines.

Related concepts

Input message formatting (Application Programming APIs)

Editing of input message segments by basic edit

The following editing is done if basic edit is used for nongraphic message data.

- For the first segment of an input message when a terminal is not in conversation mode, leading characters less than X'41' are removed. For other than the first segment or when a terminal is in conversation mode, leading characters less than X'40' are removed.

- If a terminal is not in conversation or preset mode, a left parenthesis within the first nine positions of the first segment indicates the presence of a password. The left and right parentheses and the password are removed, and the segment is compressed.
- For non-SNA devices, the X'26' character that appears as the final character in a segment is removed.
- Two asterisks (**) or two asterisks followed by NL (X'15') that appear as the final characters of a segment cause the entire segment to be discarded.
- For unbuffered keyboard devices, backspace (X'16') characters are treated as character-delete indicators. Each backspace character and the preceding input character are removed from the segment.
- If the destination of the input message is a transaction, an NL (X'15') character appearing at the end of a segment is removed.
- If a device is in preset mode, the transaction code is added to the first segment.
- For input from IBM 3270 devices, the attention identifier and cursor address are removed, and all start buffer address sequences are changed to blanks.
- If the first character of any segment is a slash (/), the entire input message is treated as a command.
- If an input message is received from an NDS device, or if it is using Intersystem Communication, the data stream is handled wholly as transparent data or transparent except for edit of the transaction code and password.

Controlling output devices

You can control an output device by using control characters in the second byte of the ZZ field of the message prefix (Z2).

Follow these rules:

- SLU–1:

  The Z2 field, bit X'80', should be set if the segment contains structured field data.

  The Z2 field, bit X'40', should be set if the segment is the first segment of a new LPAGE series.
- Switched devices:

  Exception: IBM 3270

  You can use the Z2 field in the message format to request that the line be disconnected after the present message is sent. This field is ignored if the output is physically sent to a device without this capability. The disconnect request is indicated using X'80' as the Z2 field value, this request is recognized if present in any segment.
- IBM 3270 printer (3270P device type):

  You can use the Z2 field, bit value X'80', to specify the presence of command code and WCC character in the data stream, which is to be used when MFS editing is bypassed.
- Printer components:

  The program can embed, in the text portion of the message, carriage return characters or new line symbols. If the output is going to a local printer (SYSOUT), the first two characters of the message can be carrier control characters.

Small buffer devices

Some terminal devices have hardware limitations of the maximum buffer size that they can process. Additional limits can be imposed by software, either in the device or in the network.

Exceeding these limits can result in an error and failure to deliver a message that is too large for the device to process. Be aware of what these limits are, and of the alternatives available to your application program.

Different device types specify the limits in different ways. For input, the SEGSIZE or BUFSIZE specification in IMS is relevant, and, for output, the OUTBUF specification. For VTAM terminals, the BIND

RU size is based on these values. IMS supports message chaining for both input and output, but some terminals do not support chaining. In this case, message length is limited by sizes.

Your application program has control over the length of the output message, either directly or through the MFS definition being used to format the message. If you need to send a message that is longer than the maximum length supported by the target device, you must break the message into multiple messages, each of which is shorter than the maximum length for the device. The DL/I PURG call must be issued by your application program to do this. IMS delivers messages in the order that your application program inserts them; this is a good solution for devices that are not sensitive to message boundaries.

Related reference

PURG call (Application Programming APIs)

"TERMINAL macro" on page 483

Several system definition keyword parameters on the TERMINAL macro are principal for defining an ISC session.

LINE macro (System Definition)

Controlling output

Most output of an IMS online system goes to the input terminal. You have several choices for creating printer output.

- Use local printing through hardware-specific device support.
- Code the application program so that it directs response to the printer component of the input terminal.
- Use the **/ASSIGN** command to assign the response LTERM to a printer component.
- Code the application program to send output (SYSOUT) to an LTERM representing an IMS system printer.
- Code the application program to send output to an LTERM representing an alternative device for offline printing.

The choice is dependent on the application design. The use of the **/ASSIGN** command and the control of spool data sets are choices that might involve master terminal operator intervention.

With ETO, the output is associated with the user. The user can move from terminal to terminal. The output follows the user. When the output is available, use an autologon facility to generate a session for output delivery to a specified terminal.

Using a printer component

The printer component for a terminal is often not in continuous operation. The printer might need a remote operator to supervise the paper supplies or special forms.

The **/COMPT** command is used to make the device *ready* or *not ready* if the remote operator is a VTAM component. The operator can use the **/ASSIGN** command in order to direct output to a particular component.

Related reference

/COMPT command (Commands)

/ASSIGN command (Commands)

Spooled output control

You need to be aware of requirements for IMS spooled output. This is because the output might be needed immediately by the end user or additional output can be impeded if allocated space is exhausted.

Given a spool line group, you need to be aware of how many data sets are used for output. A recommended definition is at least two data sets, one data set printing while the other receives additional input. The line number and PTERM references are also required.

At any time, you can use the **/STOP LINE n PTERM nn** command to call for the spooled output to be scheduled to a printer.

The action of this command is to close the data set and direct additional output to the next spool data set. If this is the last in the set, output goes to the first. If only one output data set exists at the time the print utility is scheduled, all messages for the LTERM are queued. This could rapidly add to the contents of the message queues. Message queues can fill very quickly, specially when the spool is the secondary master console. The IMSWTnnn procedure to execute the print utility is scheduled automatically by the data-set-full condition. A write error for the data set also starts the procedure.

The printing procedure is tailored to the line group data sets by system definition. You can invoke this procedure at any time by using a **/START REGION IMSWTnnn** command. If the data sets are not closed, current line output is queued.

Restart can affect spooled output handling. If output data sets were not printed from a prior execution, the **/NRESTART** command prevents additional output messages. The action of a **/START LINE** automatically schedules the IMSWTnnn procedure and frees the data set for output from the current system execution.

Using printer components of the IBM 3270 Information Display System

You can define printer components to be part of an IBM 3270 Information Display System. This allows for printed copy of the video output (or input) to be sent to the printer's component.

This support covers the following printer component devices: IBM 3284, 3286, 3287, 3288, or 3289.

Your system definition input indicates whether these are to be connected through a polled-BSC or an SDLC line. The printed copy can be automatically produced or be operator controlled. The input message definition or the control prefix of the output message can cause this automatic printing. The component must be attached to the same 3270 control unit as the display workstation containing the information to be copied—an IBM 3271/3274 or a 3275/3276.

Restriction: A locally attached 3270 does not support the copy function.

Specifying candidate printers

With BSC 3270, using VTAM, requests for copying the screen content to a printer are directed to the first available printer on the same control unit as the screen.

Definition: *First available printer* refers to the sequence of TERMINAL macros for the devices. A predetermined sequence controls the order in which the printer components are selected for message output. The following figure shows two display workstation printer groups.

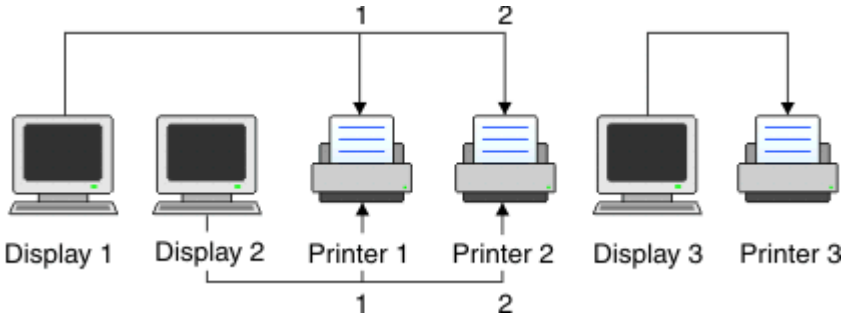


Figure 55. Sequence of IMS TERMINAL macros

The first printer defined after a display is a candidate printer for that display. Any subsequently defined printer is also a candidate if no display is defined between it and the first candidate. In the preceding figure, if a request is made for a printed copy of Display 1 or Display 2, IMS sends a 3270 copy command to Printer 3, if it is available. If Printer 3 is not available, IMS considers Printer 4. Printer 6 is not a candidate for Display 1 or Display 2, but it is a candidate for Display 5.

The copy function is not permitted to cross 3270 control-unit boundaries. For IBM 3274 and 3276 with SNA protocols, IMS sends the print request to the display, and the printer is selected by the controller. The printer need not be defined to IMS.

Restriction: The 3270R (non-SNA) copy function is not supported for ETO terminals.

Operational considerations

When a request for a copy operation is sent by the operator or an application program, the first available candidate printer is used for output.

The search order corresponds to the system definition sequence. If a printer is stopped, already printing a message, in exclusive status, or not ready, the next candidate is chosen. If a copy request by an operator finds all printers busy, the keyboard is locked until a printer becomes available. Output messages that require the copy function are not sent if a candidate printer is unavailable. The display workstation receives an error message. A retry attempt is made for the message when the error message is cleared from the screen using the message advance function (PA2 key). The format description for a message can also specify a copy action. MFS uses the DSCA operand in the DEV statement to specify a copy action.

Related concepts

System control area (SCA) and default SCA (DSCA) (Application Programming APIs)

Related reference

DEV statement (System Utilities)

Sharing printers between systems

If you plan to share the use of a printer other than the master, IMS allows an online IMS system and another subsystem (possibly another IMS system) to alternate their use of the printer.

The IMS support is for VTAM 3270 printers and, in the SNA environment, for LU–1 and LU–4 printers. The **/OPNDST NODE Q** command simulates a logon to IMS for the requested printer. The printer is automatically acquired when the owning subsystem frees the printer. You can queue output at any time independently of printer availability.

The system definition requirements are for the OPTIONS keyword of the TERMINAL macro. Specify:

OPNDST
: so that the **/OPNDST** command is valid for this terminal

SHARE
: so that the VTAM macro SIMLOGON can be issued by IMS in order to acquire the printer when data is queued for output

RELRQ
: so that IMS releases the terminal to other VTAM subsystems upon their request

Specify RELRQ on the ETO logon descriptors to have the same features available for ETO terminals.

Each output message that is queued to a shared printer attempts a session.

As an alternative to immediate logon simulation and repeated failures while trying to acquire messages, you can use the Shared Printer exit routine (DFSSIML0). This exit routine monitors terminal (or transaction) status, and takes one of three actions:

- Ignores the request
- Simulates the **/OPNDST** command
- Queues a transaction for an automated operator program

The Shared Printer exit routine is not called if the printer is connected. You specify the SIMEXIT keyword in the IMS startup parameters to call the Shared Printer exit routine. DFSSIML0 must be placed in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.

Recommendation: Do not share the secondary master terminal.

Related concepts

"Designing logical terminal networks" on page 390
The IMS system definition describes the characteristics and relationship of communication lines, static terminals, and logical terminals (LTERMs).

Related reference

Shared Printer exit routine (DFSSIML0) (Exit Routines)

Part 7. Intersystem Communication (ISC)

These topics introduce Intersystem Communication (ISC) and provide all the information you need to use ISC to connect IMS subsystems with other types of subsystems that support the ISC protocol.

© Copyright IBM Corp. 1974, 2020

Chapter 26. Overview of Intersystem Communication

ISC is a part of the IMS Transaction Manager. It is one of the ways to connect multiple subsystems. The other means of connection is Multiple Systems Coupling (MSC).

As defined under SNA, ISC is an LU 6.1 session that:

- Connects different subsystems to communicate at the application level.
- Provides distributed transaction processing permitting a terminal user or application in one subsystem to communicate with a terminal or application in a different subsystem and, optionally, to receive a reply. In some cases, the application is user written; in other cases, the subsystem itself acts as an application.
- Provides distributed services. Thus, an application in one subsystem can use a service (such as a message queue or database) in a different subsystem.

The IMS implementation of ISC supports connections that are managed by TCP/IP (ISC TCP/IP) or SNA VTAM (ISC VTAM). Both dynamically defined ISC nodes and statically defined ISC LU 6.1 terminals can use either TCP/IP or VTAM support.

ISC TCP/IP supports connections between IMS and CICS only. ISC TCP/IP support uses a private protocol, IP interconnectivity (IPIC), that is defined by CICS. The use of IPIC is generally consistent with the protocols that are defined by SNA for ISC VTAM and is transparent to application programs that use ISC.

ISC VTAM supports communication between unlike subsystems and includes SNA-defined session control protocols, data flow control protocols, and routing parameters. The functions provided by each of these protocols and parameters are summarized below, and the IMS support for them is described in this topic and its subtopics.

Session control (SC) comprises:

- Initiating sessions between subsystems
- Recovering and resynchronizing sessions, maintaining the integrity of session states and recoverable resources across both session and subsystem failures
- Terminating any or all session paths between IMS and another subsystem

Data flow control (DFC) includes:

- Controlling send and receive protocols within a session.
- Resolving contention for transaction initiation within a session.
- Monitoring error recovery processing.
- Monitoring symmetrical shutdown
- Controlling synchronization of resources. Sync point control ensures that all resources are committed or backed out synchronously.

ISC VTAM routing comprises:

- Using parameters in the SNA-defined function management headers to connect the process required for incoming messages and to route reply messages.

APPC/IMS only supports message switching to LU 6.2 destinations through the DFSAPPC service.

Related concepts

“DFSAPPC system service” on page 54
DFSAPPC is an IMS system service for exchanging messages between LU 6.2 application programs (LU 6.2 to LU 6.2), and between LU 6.2 application programs and IMS-managed LTERMs. Message delivery is asynchronous; messages are held on the IMS message queue until they are delivered.

“ETO and LU 6.1 (ISC) terminals” on page 104

© Copyright IBM Corp. 1974, 2020

For LU 6.1 (ISC) terminals, IMS supports parallel sessions to the same node name. In this case, a separate structure is built for each session. However, each session and its associated structure operate independently, as a separate terminal.

Related tasks

"ISC protocols for VTAM connections" on page 487
IMS uses ISC protocols to control sessions, data flow, and message routing over ISC VTAM connections. The following topics include the specific protocol information that you need to send and receive data with an ISC link.

Comparison of ISC and MSC

Both Multiple Systems Coupling (MSC) and Intersystem Communication (ISC) can be used to couple multiple IMS subsystems. Both MSC and ISC enable you to route transactions, distribute transaction processing, and expand beyond the capacity of one IMS system.

MSC is an IMS protocol that enables coupling of IMS systems to other IMS systems only. ISC, however, allows you to connect IMS subsystems with any other subsystem that supports the ISC protocol. This other system can be another IMS, CICS, or a user-written system.

MSC supports four types of links between IMS systems: channel-to-channel (CTC), memory-to-memory (MTM), TCP/IP, and VTAM LU 6.1. ISC uses VTAM LU 6.1 or TCP/IP; however ISC TCP/IP is supported only for connections between IMS and IBM CICS Transaction Server for z/OS.

The following figure compares MSC to ISC.

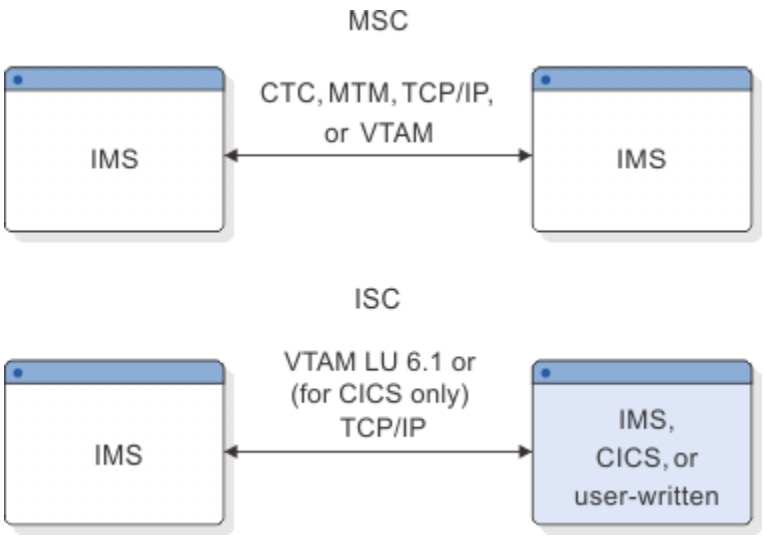


Figure 56. IMS Multiple Systems Coupling and Intersystem Communication

ISC, MSC VTAM, and MSC TCP/IP all provide parallel session support. Some key differences exist, however. The following table highlights the major functions of MSC and ISC, and shows the differences in support.

Table 76. Comparing MSC and ISC functions

MSC functions	ISC functions
MSC connects multiple IMS systems only to each other. These IMS systems can all reside in one processor, or they can reside in different processors.	ISC can connect either like or unlike subsystems, as long as the connected subsystems both implement ISC. ISC can couple an IMS subsystem to: • Another IMS subsystem • A CICS subsystem (using VTAM or TCP/IP support) • A user-written subsystem
Communication in the MSC environment is subsystem-to-subsystem.	Communication is between application programs in the two subsystems. The subsystems themselves are session partners, supporting logical flows between the applications.
Processing is transparent to the user. That is, to the user, MSC processing appears as if it is occurring in a single system.	Because ISC supports coupling of unlike subsystems, message routing requires involvement by the terminal user or the application to determine the message destination. Specified routing parameters for ISC VTAM connections can be overridden, modified, or deleted by the Message Format Service (MFS).
Unless MSC-directed routing is used, routing is automatic based on system definition parameters. The terminal operator or application program does not need to know routing information.	ISC provides a unique message-switching capability that permits message routing to occur without involvement of a user application.
MSC supports the steps of a conversation to be distributed over multiple IMS subsystems, transparent to both the source terminal operator and to each conversational step (application).	ISC connections that are supported by VTAM support MFS in IMS subsystems to assist in the routing and formatting of messages between subsystems. ISC TCP/IP connections do not support MFS.
MSC does not support the use of the Fast Path Expedited Message Handler (EMH).	When VTAM connections are used, ISC supports the use of Fast Path Expedited Message Handler (EMH) between IMS subsystems.

Related tasks

"Administering Multiple Systems Coupling" on page 685
The following topics describe the system administration activities required when you connect two or more IMS online systems in a network using MSC.

IMS facilities available to ISC

A variety of IMS facilities are available to Intersystem Communication (ISC).

Distributed transaction processing

When supporting distributed transaction processing, IMS can be either the front-end or back-end processor. A front-end subsystem originates communications traffic (transactions, commands, or message switches) that are to be processed by the back-end subsystem. A message is typically the result of data entered at a terminal. However, the message can also result from processing that occurred in an application within the front-end subsystem. The back-end subsystem processes input messages from another subsystem.

In the most typical configuration, IMS is the back-end system and processes IMS transactions that another subsystem enters.

In addition, the Front-End Switch exit routine provides special support for front-end or back-end system utilization when ISC VTAM is used. ISC TCP/IP does not support the Front-End Switch exit routine.

Sometimes, transactions entered into an IMS front end are sent with an ISC message switch to the other subsystem. In this case, the terminal operator or MFS provides the message routing information. The transaction can also be sent as alternate program communication block (PCB) output, in which a user-written application program or MFS format definition provides the routing information.

The fact that IMS is a queued subsystem is a key factor in determining the ISC functions that it supports, particularly when IMS acts as a front-end processor. Messages created by a terminal operator or an application in a front end IMS are queued for transmission to the receiving subsystem and sent asynchronously with respect to the terminal or application that sent the message. A terminal attached to an IMS front-end is not held in response mode during the receiving (back-end) subsystem's processing, unless it uses both ISC VTAM and the Front-End Switch exit routine (DFSFEBJ0).

Distributed services

IMS supports distributed services by providing remote access to an IMS message queue. Although IMS does not support distributed DL/I calls, a DL/I database in one subsystem can be updated by another subsystem; this action requires the sending subsystem to invoke an updating application in the receiving subsystem.

IMS transaction types

IMS supports the following transaction types:

Recoverable-update (includes Fast Path)
Recoverable-inquiry
Irrecoverable-inquiry

If you are using ISC with TCP/IP, some limitations can apply to the ISC support for the transaction types in the preceding list. For more information, see “Functions available to an ISC TCP/IP session” on page 589.

IMS execution modes

IMS supports the following execution modes:

Response mode (includes Fast Path)
Conversational mode
Exclusive mode
Transaction preset mode
Test mode
Non-response or non-conversational mode

If you are using ISC with TCP/IP, only non-response or non-conversational mode is supported. For more information, see “Functions available to an ISC TCP/IP session” on page 589.

IMS editing facilities

Messages transmitted on the ISC session and processed within an IMS subsystem are edited by the following editing facilities:

ISC edit (the default editor)
Message Format Service (MFS)
Basic edit

These editors can be selected on a message-by-message basis.

ISC message integrity

Message integrity is provided to prevent loss or duplication of input or output messages between IMS and another half session during session restart.

Message integrity and recovery are increased by log write-ahead (LWA). This facility is invoked during system definition by an option on the TRANSACT macro. LWA ensures that sync point information is written to the log (and thus available to IMS restart procedures) before IMS acknowledges the message. More information on message integrity can be found in the Resynchronizing sessions and Sync point and response requirements topics.

ISC security

IMS security facilities control access to IMS resources by another subsystem. Before implementing ISC security in IMS, examine the data protection facilities of the subsystems and associated operating system components. Some important ISC-environment security factors are described there.

Terminal operator verification and authorization are provided by the subsystem controlling that terminal connection.

Both IMS terminal and password security can be defined for static ISC terminals by using resource access control facility (RACF). If password security is defined, a password must be provided with the input message.

ISC in an XRF complex

ISC VTAM sessions are handled as class-2 or class-3 terminals. If the active IMS system fails, all LU 6.1 sessions are terminated. The alternate (now active) IMS system reestablishes the sessions after it takes control of the complex. Users might be without service for a short time.

ISC TCP/IP sessions are not supported by XRF.

Related concepts

“Establishing a connection with an XRF complex” on page 489
An XRF complex can be a session partner.

“Planning for security” on page 401
To prevent unauthorized use of a terminal in the IMS network, you can use RACF (or an equivalent product).

“ETO and LU 6.1 (ISC) terminals” on page 104
For LU 6.1 (ISC) terminals, IMS supports parallel sessions to the same node name. In this case, a separate structure is built for each session. However, each session and its associated structure operate independently, as a separate terminal.

“Resynchronizing sessions” on page 491
To maintain the integrity of recoverable resources, messages, and queues in IMS across both subsystem and session failures, both half sessions must maintain the session information required for the resynchronization process.

“Relationship of ISC and IMS execution modes” on page 461
Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

Extended Recovery Facility Overview (System Administration)

IMS security (System Administration)

Related tasks

“Editing and formatting IMS messages” on page 425
IMS uses two methods to edit and format messages to and from terminals: Message Format Service (MFS) and basic edit routines.

“IMS transaction types and transaction states” on page 421

Transactions are the most common type of data that is sent from a logical unit to IMS.

Related reference

“Sync point and response requirements” on page 508
The IMS input/output message flow can be represented as an input/output flow from a sequential queue data set.

Front-End Switch exit routine (DFSFEBJ0) (Exit Routines)

Sample system configurations

These figures illustrate sample IMS Intersystem Communication (ISC) configurations. The first two figures use IBM CICS Transaction Server for z/OS as an example of an ISC node.

Existing or new IMS transactions invoked from CICS

The following figure illustrates ISC's distributed transaction processing capability—the ability to invoke existing or new IMS transactions from a CICS application or from a CICS application on behalf of a terminal attached to CICS (using transaction routing). The links in the figure can be either ISC TCP/IP links or ISC VTAM links.

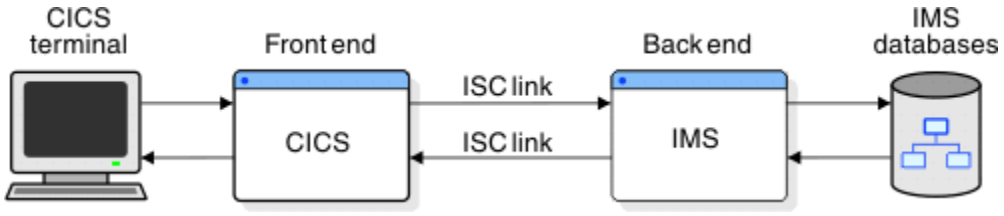


Figure 57. Existing or new IMS transactions invoked from CICS

In figure above, ISC provides transaction-to-transaction capability, because an application program can be written as two complementary transactions: one executing in an IMS system, the other in a CICS system. User functions can be distributed between systems as required. If ISC TCP/IP links are used, only asynchronous CICS transactions can be used and other limitations apply.

IMS MFS DPM mapping function distributed to CICS's BMS

The following figure illustrates ISC VTAM's ability to distribute device mapping function from an IMS Message Format Service (MFS) system to a CICS Basic Mapping Support (BMS) system. MFS partially maps the data stream that is sent to a CICS application. The application is responsible for processing the input data stream to a form that is acceptable to BMS. BMS can then be used to complete the device mapping.

MFS does not support ISC TCP/IP communication.

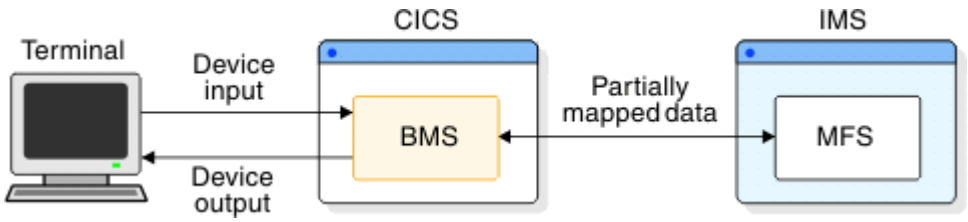


Figure 58. IMS MFS DPM mapping function distributed to CICS's BMS

IMS-to-IMS with ISC VTAM

The following figure illustrates how two IMS systems can be connected using ISC VTAM links. Interconnection of two IMS systems allows a new or modified transaction (TRAN1) in one IMS system to invoke an existing transaction (TRAN2) in another IMS system. Using MFS, replies can be routed to a

new transaction (TRAN3) in the initiating IMS system, or to a new instance of the originating transaction (TRAN1). Without MFS, replies are treated as message switches and routed to the source terminal.

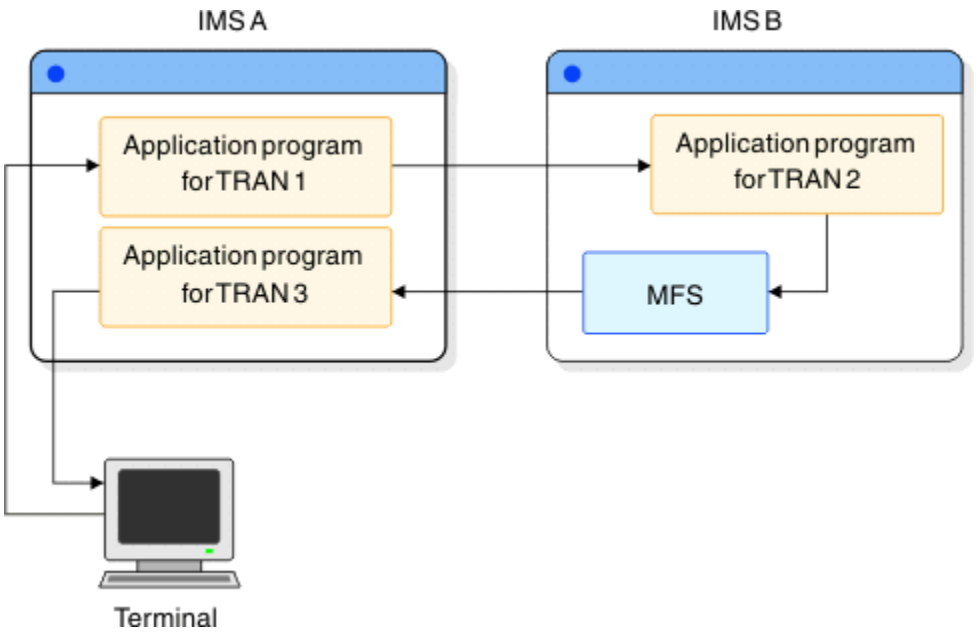


Figure 59. IMS-to-IMS with ISC VTAM

CICS-to-IMS MSC using ISC

The following figure illustrates how a CICS subsystem can communicate with an IMS subsystem that has MSC links to other IMS subsystems. In this case, MSC is used to couple the IMS subsystems, while the CICS-to-IMS session is supported by ISC. The CICS subsystem has a single-system view of the multiple IMS subsystems with which it can communicate. MSC distributes the load between the multiple IMS subsystems.

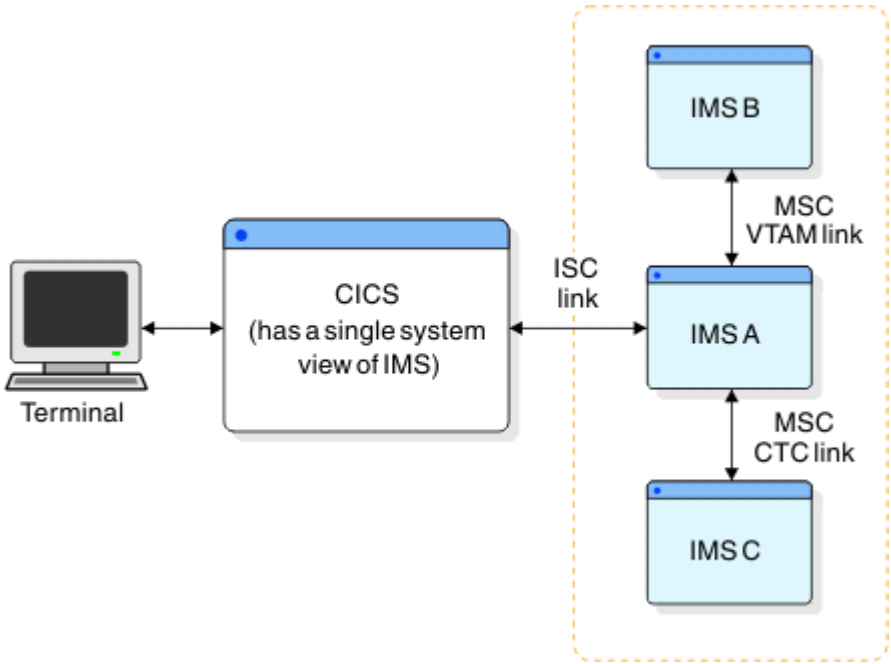


Figure 60. CICS-to-IMS MSC using ISC

Related reference

"Sample program for IMS-CICS ISC" on page 657

The following topics provide a sample program that illustrates the use of ISC between IMS and CICS.

ISC support for TCP/IP

TCP/IP can be used to support ISC communications between IMS and CICS subsystems.

ISC support for TCP/IP connections provides an alternative to the connection support provided by the Virtual Telecommunications Access Method (VTAM). Existing statically defined ISC terminals can be enabled for TCP/IP support by coding the ISCTCPIP parameter in the DFSDC*xxx* PROCLIB member. ETO logon descriptors can be used to define dynamic ISC terminals that support TCP/IP.

To communicate with CICS over an ISC TCP/IP connection, IMS uses the private CICS IP interconnectivity (IPIC) protocol for session control and data flow. However, applications programs are not sensitive to the use of IPIC or the use of either ISC TCP/IP or ISC VTAM. Unless your application programs use functions that are not supported by ISC TCP/IP, you can switch from ISC VTAM links to ISC TCP/IP links without changing application programs on either side of the link.

Related tasks

"IMS Connect and TCP/IP communications" on page 139
The IMS Connect function of IMS provides access to both IMS DB and IMS TM from TCP/IP-enabled environments.

"ISC communication with CICS over TCP/IP" on page 593
TCP/IP can be used to support ISC connections between IMS and IBM CICS Transaction Server for z/OS subsystems.

ISC between IMS and CICS

The following table identifies the ISC functions that are provided by CICS and IMS, and the IMS facilities that are available to users of CICS **START** or **RETRIEVE**, and **SEND** or **RECEIVE**.

Table 77. Facilities available to CICS START RETRIEVE and SEND RECEIVE USER

Function	Front-end system	Back-end system
Distributed Transaction Processing Initiate remote transaction	CICS"1" on page 453	CICS"1" on page 453
	CICS"1" on page 453	IMS non-response type
	IMS all transaction types"2" on page 453	CICS"1" on page 453
	IMS all transaction types except conversational"2" on page 453	IMS all transaction types except conversational"2" on page 453
Distributed Transaction Processing Application-to-Application Conversation"5" on page 453	CICS"3" on page 453	CICS"3" on page 453
	CICS"3" on page 453	IMS Response Conversational Fast Path
Distributed Presentation Management"5" on page 453 Using distributed transaction processing	IMS MFS map	IMS MFS map
	CICS user application"4" on page 453 IMS MFS map	IMS MFS map CICS user application"4" on page 453

Note:

1. Using CICS START or RETRIEVE commands.
2. If the transaction in the processing subsystem is a response mode or Fast Path transaction, a nonresponse transaction type is required in the originating system to handle the reply from the processing subsystem.
3. Using CICS SEND or RECEIVE commands.
4. The user application can optionally invoke BMS to aid in mapping functions between CICS and the terminal.
5. This function is not supported in ISC links that use TCP/IP.

Related concepts

"How IMS and CICS use the ISC interface" on page 587
When designing and implementing an ISC network that contains both IMS and CICS nodes you should be aware of the issues involved.

Terminal device-dependent data

ISC, when used with IMS, is a technique that allows programs in various subsystems (IMS, CICS, and user-written) to exchange messages with application programs in other subsystems. To minimize or eliminate the need to modify these programs, the messages that are exchanged should not contain device control characters that are unique to the originating or destination terminal.

The device control editing facilities of IMS (MFS) or CICS (BMS) should be used to remove these device control codes prior to the initiating program creating its ISC message. Likewise, the program that sends the response to a terminal should use the formatting service of its own subsystem to properly format the message before sending it. This way, all inter-program message exchanges can be independent of the originating and destination terminals. This means that each terminal's owning system should understand all formats for transactions that can be entered or received by terminals on that system.

Passing CICS data to IMS

Users want their front-end application programs to receive 3270 data streams from an attached terminal and to simply pass through this data stream (device control characters) to the back-end IMS.

The resulting output from IMS should then be a 3270 data stream that could be passed through the front-end application and out to the 3270.

Pass-through input to IMS

It is possible to send a 3270 data stream from a front-end application to IMS. However, IMS does not recognize the ISC session as a 3270 device and does not edit the device control characters from the data stream. Your IMS application program must provide this editing function.

Output from IMS

Similarly, IMS does not place 3270 device control characters in an output data stream that is destined for an ISC LTERM, because the ISC LTERM is not a 3270 device. Your application program must then construct a 3270 data stream to be sent to a front-end application that can, with minimal editing, send this data stream to an attached terminal.

This effort needs to be repeated for each device and each application program, and again after any change to a device format.

Chapter 27. VTAM facilities used for ISC connections

For ISC connections that are supported by VTAM, VTAM controls the physical transmission of data between IMS and a logical unit.

Restriction: The VTAM facilities discussed here do not apply to ISC TCP/IP connections.

Both VTAM applications (such as IMS) and other subsystems can be viewed as VTAM logical units. A *logical unit* is an addressable resource, such as an application program or a subsystem. A logical unit can also be a component of a general-purpose terminal system.

The VTAM concepts and facilities used by IMS that are particularly relevant to ISC include:

- Connection, disconnection, and establishing logon mode.
- Messages and responses.
- Request definite response 1 (RQD1) or request definite response 2 (RQD2) and the associated definite responses (DR1 or DR2). Note that definite response 1 and definite response 2 have been separated and redefined for LU 6.1 protocols. RQD2 requests and DR2 responses are now known as sync-point requests and responses and are functionally independent of those responses associated with DR1.
- Sequencing and chaining.
- Orderly session disconnection from stop bracket initiation (SBI) and bracket initiation stopped (BIS) commands.
- Facilities for ensuring orderly communication, including the use of brackets and change-direction indicators.
- Sequence number recovery.
- Receiving input and sending messages.
- Conditional bracket termination.
- Extended Error Recovery Procedure (EERP).
- Use of parallel sessions between ISC nodes.
- Support for both negotiable and nonnegotiable session bind parameters.

IMS also supports the class-of-service (COS) and session-outage-notification (SON) facilities.

Related reading: For more information about the communication concepts and facilities that govern data transmission between a VTAM application program (such as IMS) and another subsystem, see *z/OS Communications Server: SNA Programming*.

The ISC system programmer and system analyst must be familiar with these concepts and facilities in order to design and implement a communications interface between IMS and a remote subsystem using SNA LU 6.1 protocols.

Related concepts

“Resynchronizing sessions” on page 491
To maintain the integrity of recoverable resources, messages, and queues in IMS across both subsystem and session failures, both half sessions must maintain the session information required for the resynchronization process.

“Using SON/COS support in IMS” on page 419
Session outage notification (SON) and class of service (COS) are facilities of VTAM and SNA that allow IMS to recognize a session outage.

Related reference

“Symmetrical session shutdown for LU 6.1 (SBI and BIS)” on page 540

© Copyright IBM Corp. 1974, 2020

Two data flow control commands allow a symmetrical and orderly termination for peer level LU 6.1 half sessions: stop bracket initiation (SBI) and bracket initiation stopped (BIS).

VTAM commands and indicators

VTAM commands and indicators (communication control information) are necessary for data transmission between an IMS application program and another VTAM logical unit.

The following table shows which VTAM commands and indicators that IMS sends and receives during an IMS session (X=supported). Results are unpredictable if any unsupported commands or indicators are used.

Table 78. VTAM commands and indicators sent and received by IMS

VTAM command or indicator	IMS sends as Primary half session, Receives as secondary half session	IMS sends as Secondary half session, Receives as primary half session
Independent of session		
Initiate session		Note “4” on page 457
Procedure error	Note “1” on page 457	Note “1” on page 457
Terminate session		Note “5” on page 457
Session control commands		
Bind	X	
Bind response		X
STSN	X	
STSN response		X
Start data traffic (SDT)	X	
SDT response		X
Unbind	X	X
Normal (synchronous) flow indicators		
Begin bracket (BB)	X	X
End bracket (EB)	X	X
Change direction (CD)	X	X
Commands		
BID	Note “3” on page 457	X
CANCEL	X	X
CHASE	X	X
Logical unit status (LUSTATUS)	X	Note “2” on page 457
Ready to receive (RTR)		X
Bracket initiation stopped (BIS)	X	
Expedited (asynchronous) flow commands		
Request shutdown (RSHUT)	X	Note “2” on page 457
Signal	X	X

Table 78. VTAM commands and indicators sent and received by IMS (continued)

VTAM command or indicator	IMS sends as Primary half session, Receives as secondary half session	IMS sends as Secondary half session, Receives as primary half session
Stop bracket initiation (SBI)		X

Notes:

1. Sent by VTAM.
2. Not sent by IMS, but optionally sent by the other subsystem and received by IMS when IMS is a primary half session.
3. Not sent by IMS, but optionally sent by the other subsystem and received by IMS when IMS is a secondary half session.
4. Supported by internal VTAM forms of CINIT (LOGON exit).
5. Supported by internal VTAM forms of CTERM (NS and LOSTERM exits).

Using the VTAM application programming interface

The VTAM application program interface (API) used by IMS for ISC consists of macro instructions and control blocks.

The macro instructions and control blocks allow IMS to:

- Establish connection or disconnection.
- Request and control the transfer of data between IMS and another logical unit.

To provide application-to-application communication, the VTAM API provides both a primary and a secondary session application interface. The primary interface is essentially the same as that used by IMS to support other VTAM nodes. However, the interface introduces some new macros and some additional parameters on existing macros.

The VTAM API supports single sessions as well as parallel (multiple, simultaneously active) sessions between two logical units. The IMS interface for this support is primarily through the session qualifier fields of CINIT and BIND, which are available through the LOGON and SCIP exit routines. Single-session support requires static definition of message queues during IMS system definition; parallel-session support permits message queues to be dynamically allocated at session initiation.

For more information about VTAM, see *z/OS Communications Server: SNA Resource Definition Reference*.

Related concepts

VTAM interface considerations (System Administration)

Specifying logon modes when establishing a connection

When establishing a connection, IMS requires that session parameters define the rules a logical unit must follow when communicating with IMS. These session parameters are contained in the VTAM mode table.

The use of the default logon mode table entry can be overridden in one of two ways:

- At system definition, the logon mode entry can be specified on the TERMINAL macro using the MODETBL=keyword or an ETO logon descriptor.
- The MODETBL keyword on the **/OPNDST** command can specify the logon mode table entry to replace the table entry defined at system definition.

Related reading:

- For information on specifying the MODETBL keyword, see *IMS Version 15 System Definition*.
- For information on specifying the **/OPNDST** command, see *IMS Version 15 Commands, Volume 2: IMS Commands N-V*.

If neither method is used to specify a logon mode table entry, VTAM uses the first entry in the default logon mode table, unless it is overridden by VTAM node mode table entry definition. If a logon mode table entry is specified, VTAM searches the default or user-specified logon mode table for the specified entry.

IMS examines the set of session parameters within the indicated VTAM mode table entry and overlays only those parameters on which IMS has dependencies. The remaining bytes are not changed. IMS ignores any unformatted user data transmitted with the session parameters.

Related reading: For more information on establishing logon mode tables and defining logon mode table entries, see *z/OS Communications Server: SNA Resource Definition Reference*.

Related reference

Bind parameters for SLU P and LU 6.1 (System Programming APIs)

Macros used in IMS environments (System Definition)

/OPNDST command (Commands)

Design considerations for secondary logical units

When IMS and another subsystem establish an ISC session, one session partner is the bidder and the other the first speaker (the contention winner).

In an IMS ISC session, the bidder is always the primary logical unit (the bind sender), and the first speaker is always the secondary logical unit (the bind receiver). When providing a system design for a secondary logical unit that is to communicate with IMS within an ISC network, consider the following points:

- The first speaker (secondary logical unit) must resolve bracket contention and be prepared to receive interrupts (for example an VTAM command or bracket indicator) at all times. If the first speaker is not prepared to do so, a deadlock can result (for example, if both the primary logical unit and the secondary logical unit use the VTAM SEND macro, and issue POST=RESPONSE rather than POST=SCHED).
- When terminating a session using the VTAM TERMSESS macro, the system designer should be aware of the differences involved in issuing CONDITIONAL versus UNCONDITIONAL. Sending CONDITIONAL permits the primary logical unit to clean up and control session termination. Sending UNCONDITIONAL causes immediate termination of the session by VTAM prior to notification of the primary logical unit. If the primary logical unit is in a suspended state as, for example, when waiting for a response, the system designer should be aware that UNCONDITIONAL session termination can have unanticipated results.

The term "secondary logical unit" is synonymous with "first speaker" and the term "primary logical unit" with "bidder".

Chapter 28. IMS facilities affected by ISC

The following topics explain how IMS facilities are affected by ISC processing.

Editing messages

Within IMS, the communication interface is transparent to application programs.

In support of this, IMS provides some common editing that:

- Appends protocols and function management headers to outbound messages
- Strips function management headers from incoming messages and moves the messages to appropriate points within the IMS control blocks

Subsequent to this common editing, messages received by IMS are edited by ISC edit, Message Format Service (MFS), or basic edit in accordance with the input ATTACH or SCHEDULER DPN parameter. Messages sent by IMS are edited by Message Format Service before they are sent. The edit functions of ISC edit and basic edit are unique. However, the associated protocols are the same.

- ISC edit is the default editor. When ISC edit is requested and that request does not contain the SNA-defined PRN parameter following the input data, the message is edited with partial input data transparency. That is, only the IMS-required destination code and optional password, which must occur at the beginning of the message, are subjected to editing by ISC edit. The balance of the input message is not edited prior to processing unless the transaction is defined to translate to uppercase. Full input data transparency is provided through ISC edit using the SNA-defined PRN parameter on the ATTACH or SCHEDULER FM header. This enables the IMS transaction code to be external to the message and enables IMS to receive input and route it to its proper destination (terminal or transaction) without examining the input message itself. Because the transaction code is external to the data stream, the optional password is not available to the session. Therefore, password security should not be defined. If password security is defined, an error results.
- Message Format Service's Distributed Presentation Management (DPM) option divides responsibility for message formatting between MFS and a program residing in another subsystem. When using DPM-Bn, physical terminal characteristics are not defined to MFS. Instead, IMS sends MFS-formatted data to a program in the other subsystem. That program must complete the formatting, if necessary, and present the data to the physical device. DPM also allows user involvement in specifying input and output ISC message routing parameters. You can also use the PRN parameter on the ATTACH or SCHEDULER FM header to specify the IMS destination externally, relative to the resulting MFS-formatted input data stream.
- Basic edit provides standard editing for terminal input. It is used to edit device and operator control characters and can be used in an ISC session when it is important to maintain input compatibility for existing applications. Protocols for basic edit are the same as those for ISC edit. Use of the ATTACH or SCHEDULER FM header PRN parameter has no effect when basic edit is used. The message destination is determined from the first field of the input data stream.

IMS permits basic edit, ISC edit, or MFS to be invoked in order to handle communication of transactions, commands, and message switches. MFS DPM facilities are defined as optional on a component-by-component basis. Any of these available processes are selectable on a message-by-message basis by using the ATTACH or SCHEDULER FM header.

ISC input and output can be edited by most IMS data communication exit routines. The exceptions are the MSC-related exit routines, and the hardware-required routines.

Related tasks

“Editing and formatting IMS messages” on page 425

© Copyright IBM Corp. 1974, 2020

IMS uses two methods to edit and format messages to and from terminals: Message Format Service (MFS) and basic edit routines.

Related reference

"FM headers for message routing" on page 469
ISC message routing information is provided within SNA-defined function management (FM) headers.

"Using FM headers to invoke ISC edit" on page 542
In the following situations, IMS can use ISC edit (ISCEDT) to edit transactions, commands, and message switches between LTERMs for an IMS-ISC session

How MFS formats input messages (Application Programming APIs)

Issuing IMS commands from an ISC session

Although available to the ISC session, IMS operator commands are primarily intended for use by appropriately authorized operators of IMS master terminals and remote terminals that are directly attached to an IMS system.

Terminal operators of other (non-IMS) subsystems do not communicate directly with IMS, but rather with the control program or user-provided applications within the other subsystem. That subsystem must determine and provide procedures for its terminal operators and application programmers to follow when communicating with IMS. If an application in a non-IMS subsystem is permitted to issue IMS commands through the ISC session, IMS dependencies are introduced into that application. An ISC half session cannot be defined as the primary or secondary master terminal.

Effects on parallel sessions

Include the USER keyword when using authorized IMS commands with the NODE keyword.

Equivalent action is taken against all active, or potentially active, parallel sessions of the indicated node when a specific parallel session instance is not identified to IMS by the USER keyword.

For example, issuing the **/CLSDST NODE x** command without specifying USER schedules termination of all active parallel sessions of NODE x. Issuing the **/STOP NODE x** command without specifying USER schedules termination of all active parallel sessions of NODE x and also prevents any new parallel sessions from being initiated on NODE x.

Using IMS test mode for ISC VTAM sessions

You can test ISC VTAM communication protocols and editing facilities by putting a back-end IMS system into test mode.

Restriction: IMS test mode is not supported for ISC TCP/IP sessions.

While in test mode, IMS receives an input message, checks input protocols and FM header parameters, performs input and output editing, inserts appropriate output protocols and FM header parameters, and returns the message to the front-end subsystem. Noncommunication error analysis, transaction code verification, and transaction scheduling are bypassed. Also, IMS does not send asynchronous output while in test mode.

You can place one or all of the ISC sessions within a specific subsystem in test mode when the IMS **/TEST** command is received on the ISC sessions. The **/TEST** command can only be entered by the terminal that is being put into test mode. You can terminate test mode by sending an end-bracket (EB) indicator on an SNA **LUSTATUS** or **CHASE** command. Test mode can also be terminated when an **/END** command is received on the ISC session or is entered locally in the back-end IMS processor by an authorized terminal operator.

In an IMS-to-IMS environment, use the ISC message switch facility to place the back-end IMS system into test mode, to send the message and receive the echo reply, and to terminate test mode. This provides an efficient means of testing without requiring a user application in either subsystem.

IMS control block storage on ISC parallel sessions

The storage required for the IMS control block structure representing potential sessions is greater than the storage required for control blocks representing input and output message queues.

Installations needing only a few logical units type 6.1, each having a relatively small number of associated parallel sessions, but each requiring that a large number of ISC users (subpools or LTERM sets) be dynamically allocated to them, can have lower storage requirements. For static ISC terminals, these users (subpools) are still statically defined. The IMS control block structure for ISC single session (statically defined and allocated LTERMs) is the same as for other static VTAM terminal types within IMS.

You do not need to define a maximum number of ETO-ISC sessions. You can continue to add sessions until you run out of storage and processor capacity. Therefore, the SESSION= keyword is not one of the supported keywords on the ETO descriptor. To define the maximum number of sessions for static ISC sessions, use the SESSION keyword on the TERMINAL macro.

Related reference

"ISC data flow control examples" on page 643
The following topics provide examples of ISC data flow control.

Relationship of ISC and IMS execution modes

Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

External specification of execution modes

Messages in ISC VTAM use the SNA-defined function management headers. ISC messages that are routed by VTAM can be processed either synchronously or asynchronously, as determined primarily by the type of FM header and secondarily by the VTAM bracket protocols that are sent with the message.

Messages that use ISC TCP/IP for communication between IMS and IBM CICS Transaction Server for z/OS use the IPIC message format, which is an internal protocol that is defined by CICS. ISC TCP/IP messages can be processed only asynchronously.

The following table provides a summary of supported VTAM protocols. Using FM headers and VTAM bracket protocols to establish the mode of the message (synchronous or asynchronous) can be considered to be external to the message.

Table 79. Processing mode requested by FM headers

FM header with VTAM bracket protocol	Synchronous	Asynchronous
ATTACH with CD	X	
ATTACH with EB		X
ATTACH without either EB or CD	X	
SCHEDULER with CD	X	X
SCHEDULER with EB		X
SCHEDULER without either EB or CD	X	X

Messages that are received by IMS with the ATTACH FM header are processed:

- Synchronously, if the session state is left "in-brackets" after the message is received.
- Asynchronously, if either of the following occurs:
  - The session state is left "between-brackets" after the message is received.
  - The message is sent with EB on the first- or only-in-chain.

During synchronous processing, unsolicited or asynchronous output is not sent.

All messages that are received by IMS with the SCHEDULER FM header are processed asynchronously by the receiver with respect to the session.

Related concepts

"Handling IMS response mode or conversational output errors" on page 505
This topic describes how IMS handles response and conversational mode errors during an ISC session and how to keep the half sessions in sync.

Related reference

"Function management headers" on page 541
In SNA, function management (FM) headers are an optional part of the request unit sent over a link. This topic describes the FM headers supported by IMS on ISC sessions.

Internal definition of execution mode

The IMS internal execution mode determines how IMS processes a transaction.

Response mode and conversational mode, for example, are used to ensure synchronism between a given input message and its associated reply. Although a transaction uses an IMS input and output message queue between the session and the application, these modes ensure that associated messages are always processed synchronously with relation to the source session. While these messages are being processed, IMS does not send asynchronous or unsolicited output on that session.

Internal IMS definitions are either synchronous or asynchronous.

The following modes are synchronous. Except for certain commands, ISC TCP/IP sessions not support these modes:

- Response mode
- Conversational mode
- Fast Path
- Commands
- Test mode

The IMS commands that ISC TCP/IP sessions support include **/DISPLAY** and **/RDISPLAY**.

The following modes are asynchronous:

- Nonresponse mode
- Nonconversational mode
- Message switch

Resultant processing mode during ISC VTAM communications

During ISC VTAM communications, when the relationship between the externally requested execution mode and the internally understood processing mode are consistent, the message is processed exactly as requested. In the case in which the two specifications (internal versus external) are not consistent, the message's execution with respect to the session is synchronous.

In either case, if the generated reply is returned on the same session as the request, it is sent using the same type of FM headers that were received with the input message. If the reply is returned on a session other than the one on which the input message was received, it is considered unsolicited asynchronous output and is sent with the SCHEDULER FM header. If the subsystem that is to receive the reply does not support receipt of the SCHEDULER header, the reply is sent with an ATTACH header that terminates the bracket at the end of the IMS message. Therefore, except for a nonlast conversation, nonlast MFS demand-paged messages, or test mode output, ATTACH is used instead of ATTACH SCHEDULER and EB is indicated on the first or only chain of the message. A Nonlast conversation, nonlast MFS demand-paged messages, and test mode output are sent carrying ATTACH with CD indicated on the first or only chain of the message.

However, when a message is sent from the front-end subsystem to the back-end subsystem that is externally requesting synchronous processing, no assumptions should be made by the front-end subsystem as to the timing of the output reply or the availability of any other output for that session. As a result, requests and replies cannot be correlated within the front-end subsystem, even though execution in the back-end subsystem might be synchronous. This is summarized in the following table.

The special support for front-end and back-end system utilization Front-End Switch exit routine provides.

Traffic between two IMS systems is always sent in asynchronous format. If the internal definition of the transaction within the back-end system is synchronous, it is processed synchronously and the ISC session is held for synchronous output. However, this same relationship does not apply for the source terminal. The front-end IMS does not hold that terminal in response mode until the reply is received from the back-end subsystem, unless you are using the Front-End Switch exit routine.

Table 80. Internal versus external execution mode specification

Internal specification (system definition)	**External specification (FMH)**	
	Synchronous"1" on page 463	**Asynchronous**"2" on page 463
Synchronous	Synchronous	Asynchronous"3" on page 463
Asynchronous	Synchronous"4" on page 463, "5" on page 463	Asynchronous

Notes:

1. Synchronous specification does not include ATTACH with EB indicated on the first or only chain of the message.
2. Asynchronous specification includes ATTACH with EB indicated on the first or only chain of the message.
3. Synchronous operation in IMS is allowed in this case, because the SNA-defined external asynchronous format allows IMS as the message receiver to execute an inbound message according to its own interpretation. Thus, IMS's internally synchronous definition overrides the external asynchronous format. If, however, IMS's internal (system definition) mode of execution is synchronous and the message is received with ATTACH EB, IMS generates an error message and terminates the session.
4. This support allows response mode to be established dynamically (on a message-by-message basis), even though the message was internally defined to be asynchronous (nonresponse mode).
5. Messages cannot be externally defined as synchronous when OPTIONS=NORESP is specified internally on the IMS TERMINAL macro or ETO user descriptors. This conflict between system definition and FM header intent causes session termination.

Traffic between two IMS systems is always sent in asynchronous format. If the internal definition of the transaction within the back-end system is synchronous, it is processed synchronously and the ISC session is held for synchronous output. However, this same relationship does not apply for the source terminal. The front-end IMS does not hold that terminal in response mode until the reply is received from the back-end subsystem, unless you are using the Front-End Switch exit routine.

Related reference

Front-End Switch exit routine (DFSFEBJ0) (Exit Routines)

LTERM users (subpools) and components

IMS user blocks are sets of IMS logical terminals (LTERMs) defined by the SUBPOOL macro during IMS system definition or dynamically created from ETO user descriptors.

Subpools defined at system definition cannot be used with ETO LU 6.1 terminals. Users (subpools) defined for ISC are separate from the users defined for dial-type terminals and are permitted only in conjunction with ISC parallel session support.

Definition: The collection of all static ISC users is known as the *VTAMPOOL*. LTERMs defined within the VTAMPOOL are reassignable only between users within the VTAMPOOL. LTERMs not defined within the VTAMPOOL cannot be assigned into the VTAMPOOL.

ISC users are dynamically assigned to an ISC session instance as a result of session initiation. That is, these parameters define the user that is available to the given session instance. This user remains allocated to the named parallel session instance even across session and subsystem failures until released through normal termination by mutual agreement of IMS and the other subsystem. For a single nonparallel session, the allocation of a set of LTERMs is fixed during system definition.

Each IMS LTERM is associated with one input and one output IMS component. The input and output components can be the same component, or different components can be specified. Conversely, IMS does not prevent multiple input or output LTERMs from being associated with a single component. However, doing so can cause problems with input component determination or output presentation.

IMS uses the input component ID to identify the LTERM that is to be associated with the input message. For other terminal support, IMS assumes that all input is from the first LTERM in the list that passes the necessary operational and security checks. However, for input from an ISC node, the input component is determined based on the component that is indicated in the ATTACH FM header. If no FM header is received, IMS assumes the input is to be associated with the LTERM for input component (ICOMPT) one. After the component value is determined, if the associated LTERM cannot be found, is stopped, or is not ready, or if that LTERM cannot pass security checks, the message is rejected.

When output is sent, it is sent to the component (COMPT) identified with the output LTERM. Message switches, broadcast messages for specific LTERMs, and data replies from transactions are directed to the component that is associated with the specified output LTERM. The user-written MPP can insert to the I/O PCB and default to the output component that is associated with the selected input LTERM. It can also address specific components through the appropriate LTERM name by an insert to an alternate PCB.

You can establish proper relationships between input and output components through the NAME macro during IMS system definition or LTERM keyword on the ETO user descriptor. This enables a logical unit to indicate its input component and causes output to be returned to the associated output component that was indicated during IMS system definition or on an ETO user descriptor. Proper definition and use of components can reduce or eliminate the need for LTERM naming conventions, DL/I CHNG calls, and inserts to alternate PCBs.

Recommendation: Incorrectly specifying the message delete system definition parameter, MSGDEL, or the ETO user descriptor while defining ISC users (subpools) can prevent a session from being initiated. An ISC session can be initiated only if the MSGDEL specifications on the TERMINAL or ETO user descriptor and SUBPOOL macros match.

Also, when using the ISC message switch support or an alternate PCB insert from a local transaction in order to route messages to another subsystem and return replies to a source terminal operator, MSGDEL=SYSINFO must be specified on the IMS system definition TERMINAL macro or ETO logon descriptor associated with both the ISC session and the source terminal. Specifying MSGDEL=NONIOPCB for the ISC session prevents ISC-message routing to other subsystems. Specifying MSGDEL=NONIOPCB for the source terminal prevents message replies from being routed to the source terminal operator.

The output bracket and send and receive protocols used by IMS depend on a combination of the system definition output component specifications, the protocol used to input the message to IMS, and the type of IMS message received. For example, IMS synchronous message types have a predefined protocol for output replies. Associated with the output component is a definition of send and receive protocols for asynchronous output.

Related concepts

“IMS use of routing parameters” on page 470
IMS ISC without the Front-End Switch exit routine does not provide for automatic transaction routing when IMS is the front-end subsystem.

Defining VTAM terminals (System Definition)
ETO descriptors (System Definition)

Related reference

“Bracket protocol for IMS output” on page 518
The output bracket and send/receive protocol used by IMS and the number of output messages sent per bracket are dependent on a variety of factors.

VTAMPOOL macro (System Definition)

SUBPOOL macro (System Definition)

TERMINAL macro (System Definition)

Chapter 29. Designing communications using the ISC protocol

The following topics provide an overview of subsystem-to-subsystem communications that use the ISC protocol.

Determining output protocols

Traffic on the ISC session is controlled by the VTAM protocols associated with the message.

The primary bracket protocols used are:

- Begin bracket (BB)

  Signals the beginning of a bracket. Begin-bracket is an unconditional request when issued by the first speaker (secondary half session) and a conditional request when issued by the bidder (primary half session).

- End bracket (EB)

  Signals the end of a bracket, places the session in contention state, and allows either session partner to request a new bracket.

- Change-direction (CD)

  Turns control of the session over to the session partner and allows the session partner to send traffic on the session.

Traffic across an ISC session can be sent and received synchronously to the session. The processing of messages is performed either synchronously or asynchronously to the session. That is, the sending subsystem initiates a message and waits for a reply (synchronous) or does not wait (asynchronous).

Within ISC, the external synchronous mode of operation (as specified by FM headers) requires transmitted messages and associated replies, if applicable, to occur within a single bracket. The end of synchronous mode is signaled by EB. When in synchronous mode, IMS must be able to reply, if required, within the same bracket (before EB is received) to any outstanding message traffic with the exception of a message switch. If the input message protocol prohibits IMS from sending any required replies within the same bracket, the input message is rejected.

Within IMS, the protocols used for output that must be sent with ATTACH (because the other subsystem lacks SCHEDULER support) are predefined, regardless of whether the originating input transaction and resulting output replies are synchronous (ATTACH) or asynchronous (SCHEDULER). This output includes:

- Nonlast chains (pages) of MFS paged output
- The last (MFS paged) or only chain of response mode, conversation mode, test mode, and IMS command replies
- Asynchronous output

You must define to IMS the protocols to be used with the last (MFS paged) or only chain of other asynchronous output. These are defined on the COMPTn keyword of the TERMINAL macro or on an ETO logon descriptor. Four parameters are supplied:

SINGLE1
Asynchronous output for this component is sent one message per bracket.

SINGLE2
Asynchronous output for this component is sent one message at a time with the VTAM begin-bracket (if necessary) and change-direction indicators to allow the receiving subsystem to optionally send its communication traffic.

MULT1
All asynchronous messages for a given LTERM are sent before the bracket is ended.

© Copyright IBM Corp. 1974, 2020

MULT2
All asynchronous messages for a given LTERM are sent before change-direction is sent.

For IMS-to-IMS sessions, the selection of SINGLE1, SINGLE2, MULT1, or MULT2 can be related to the transaction types and characteristics of the receiving IMS subsystem. The following considerations apply:

SINGLE1
This protocol is appropriate for sending message switches, nonresponse transactions, or nonconversational transactions to the receiving IMS.

SINGLE2
This protocol is appropriate for sending response mode (including Fast Path) transactions, commands, and test mode, because synchronous communication assumes that input generates corresponding output. Conversational mode is not supported on an IMS-to-IMS session.

MULT1 or MULT2
These protocols treat asynchronous traffic the same way as SINGLE1 or SINGLE2 treat synchronous traffic. They are useful in suppressing contention when large amounts of asynchronous traffic are queued for transmission on one (MULT1) or both (MULT2) subsystems.

Related concepts

"Relationship of ISC and IMS execution modes" on page 461
Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

Related reference

"Bracket and half-duplex protocol" on page 515
IMS uses SNA bracket protocol to resolve contention, and uses the change-direction indicator of the half-duplex protocol to control normal flow send/receive mode while within bracket state.

"Bracket protocol for IMS output" on page 518
The output bracket and send/receive protocol used by IMS and the number of output messages sent per bracket are dependent on a variety of factors.

Accessing existing application programs with ISC

During an ISC session, you can access application programs written prior to your installation's support of ISC.

Consider the information in the following topics when planning to access these programs.

Accessing programs that use MFS

If a program that uses MFS program is sensitive to the MFS MODname, MFS must be used when input is received from an ISC session.

Your program is sensitive, for example, if the program tests the MODname in the IOPCB as part of its logic, or uses ISRT calls to the IOPCB that include a MODname as part of the call.

If your program is sensitive to cursor position, it cannot be used to receive ISC session input. An ISC LTERM is not a hardware device and, as a result, cursor position has no meaning.

For output, MFS provides support for attribute and extended attribute bytes.

ISC supports the system control area (SCA) and default system control area (DSCA) in the following manner. All functions allowed for the 3270 display can be specified by the application program in a message field (MFLD) defined as an SCA. A device field (DFLD statement) can be defined as an SCA in the device output format (DOF). For ISC subsystems, MFS does not interpret the SCA specifications; it merely relays them in the user-defined device field SCA that it sends to the remote program or ISC subsystem. The MFS-supported SCA, or DSCA, for ISC session output can be used by the receiving application program to intelligently handle a terminal device attached to its subsystem.

If your application program is not sensitive to MFS formatting, it is possible for an ISC-connected subsystem to send an input data stream to your program without invoking MFS. To accomplish this,

the sending subsystem must construct an input data stream that is exactly the same as your program's I/O area. This "bypassing" of MFS would result in a reduction of the processor overhead associated with MFS. Bear in mind that bypassing MFS in this manner can result in the transmission of inefficient data streams from the sending subsystem and also makes the sending subsystem sensitive to changes in your program's I/O area.

It is probably easiest, from an implementation point of view, to use MFS with ISC when MFS is already used for input to your program. In this case, you simply create ISC (DPM-Bn) device input formats (DIF) and DOFs.

Related reference

DFLD statement (System Utilities)

Accessing programs that do not use MFS

The two input edit choices for existing application programs that do not use MFS are ISC edit and basic edit.

ISC Edit is the default edit process for ISC input to IMS. An existing application program expects the transaction code to be at the start of the input message and to be followed by a blank. When this is true, the sending subsystem should ensure that the transaction code is placed at the start of the data stream, followed by a blank. Password security is for statically defined terminals only and can only be invoked by placing the password after the transaction code in the data stream and ensuring that the SNA-defined PRN in the FM header is not present. When ISC Edit is used, the input data stream sent to IMS should match the application program's I/O area exactly.

Basic edit for ISC input provides standard editing as if the input comes from a terminal. It is used to edit device and operator control characters. The sending subsystem could simply pass through to IMS a data stream from a device that would normally use basic edit if it were attached directly to IMS. In this situation, IMS's ISC basic edit support eliminates application-provided editing in the sending subsystem. Basic edit is selected by the sending subsystem when it places the characters 'BASICEDT' in the DPN field of the FM header.

Routing messages

The following topics introduce the routing fields in the SNA-defined FM headers and describes how IMS uses these fields to route messages (and thus support distributed processing).

FM headers for message routing

ISC message routing information is provided within SNA-defined function management (FM) headers.

On input, IMS automatically processes the information in these FM headers and then removes them from the input message. IMS then passes the message to the appropriate input edit process. On output, IMS automatically builds the necessary routing FM headers based upon information provided by IMS system definition, session bind parameters, the input FM header, or MFS.

For this explanation of distributed transaction processing, the portion of the ATTACH and SCHEDULER FM headers of interest are the routing parameters that they carry. Four fields are used to route messages to appropriate control blocks. These routing fields, present in both headers, are:

DPN (Destination Process Name)
The DPN parameter names an input process to be invoked synchronously to the session. Within IMS, the processes that can be invoked are basic edit, ISC edit, or MFS. For MFS, the input DPN value is the MFS message input descriptor (MID) name. For other subsystems, the named process can be the message destination (such as a transaction code).

PRN (Primary Resource Name)
The PRN parameter names a resource to be associated with the attached process. This parameter occurs on a reply returned as the result of processing a message in a remote subsystem.

Within IMS, the input PRN can name either an LTERM message queue for terminal output (IMS message switch) or an input transaction message queue. When the PRN is not available on input, IMS

defaults to the normal method of using the message destination contained in the first data field (or optionally inserted by MFS formatting) or a preset transaction code established by an IMS operator command.

RDPN (Return Destination Process Name)
The RDPN parameter defines a suggested return destination process name to be associated with an output message. It should be returned as the DPN on resulting message replies to facilitate the processing of replies returned to the source subsystem. The input RDPN is discarded when replies or message switch output is sent on another session. The RDPN is sent by IMS only for MFS-formatted output, wherein the RDPN is the optional chained MID name specified on the MFS message output descriptor (MOD).

RPRN (Return Primary Resource Name)
The RPRN parameter defines a suggested return primary resource name to be associated with an output message. It should be returned on resulting message replies to the subsystem that was the source of the message to facilitate return reply processing within that subsystem. The input RPRN is discarded when replies or message switch output is sent on another session. The RPRN can optionally be set by use of MFS formatted output if specified on the MFS MOD.

Related reference

“Function management headers” on page 541
In SNA, function management (FM) headers are an optional part of the request unit sent over a link. This topic describes the FM headers supported by IMS on ISC sessions.

“ATTDPN” on page 561
The destination process name (ATTDPN) parameter identifies, explicitly or implicitly, the input process to be attached to the half session.

“ATTPRN” on page 562
The ATTACH primary resource name (ATTPRN) parameter represents the destination for an input message in the receiving subsystem.

“ATTRDPN and ATTRPRN” on page 563
The return destination process name (ATTRDPN) and return primary resource name (ATTRPRN) parameters define the reply process and the return primary resource within the source session and should be returned to the source session on resulting replies to facilitate return-reply routing within the source session.

“ATTDQN and ATTDP” on page 564
The destination queue name (ATTDQN) parameter names a specific message instance. This parameter is valid only for output MFS demand-paged messages.

IMS use of routing parameters

IMS ISC without the Front-End Switch exit routine does not provide for automatic transaction routing when IMS is the front-end subsystem.

These transactions can be sent as alternate PCB output from a user-written application program within IMS or by using an ISC message switch. In the latter case, the source terminal operator provides the output routing information through an ISC destination LTERM name which is the required first data field on IMS message switches.

Except when the output is formatted by MFS (by using a MOD chained from the input MID associated with the message switch), IMS message routing support for ISC removes the first data field before sending the balance of the message text to the back-end subsystem. When MFS is used to format message switch output, you should use MFS editing to remove the output LTERM name.

Further, ISC message routing support within IMS defaults the RPRN within the output FM header sent with the message to the LTERM name associated with the terminal that was the source of the message switch. For alternate PCB output, the RPRN is set to the I/O PCB name. The RPRN allows IMS to automatically route message replies from the remote subsystem back to the source terminal in the form of a message switch, provided the other subsystem has returned the RPRN parameter as the PRN parameter on the reply. MFS DPM can be used to alter the FM header destination and return routing parameters. When

using IMS as a front-end subsystem to route messages to another subsystem or route replies back to the source terminal, MSGDEL=SYSINFO must be specified on the IMS system definition TERMINAL macro statement or an ETO user descriptor associated with both the ISC session and the source terminal.

Routing examples

The following routing examples show typical functions that can be accomplished between two subsystems using ISC and how routing parameters are used to accomplish these functions.

In the examples:

- The ISC edit alias is ISCE.
- The LTERM name for input and output associated with the terminal on the left in the diagrams is T.
- The LTERM name for input and output associated with the terminal on the right in the diagrams is T2.
- The LTERM name for the ISC session between the two subsystems is LTISC1.

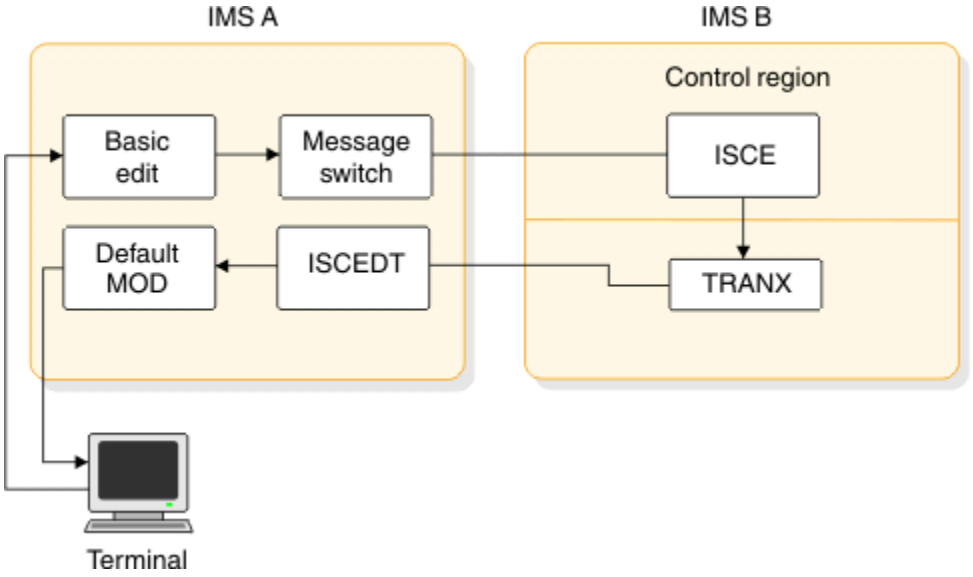
Related reading: For an example of the Front-End Switch exit routine, see *IMS Version 15 Exit Routines*.

Related reference

Front-End Switch exit routine (DFSFEBJ0) (Exit Routines)

Example 1. IMS-to-IMS message switch routing

The following figure shows the terminal attached to IMSA uses the ISC message switch to input TRANX to be executed in IMSB.



Note to figure: The name ISCE was defined for ISC edit during system definition by specification of EDTNAME=ISCE on the COMM macro in both subsystems.

Figure 61. ISC example for IMS-to-IMS message switch routing

An MPP in IMSB processes TRANX, and the reply is routed back to the terminal in IMSA, which was the source of the message switch. The output terminal reply appears as an output message switch.

1. Assume the terminal is a 3270 display device. Based on this assumption, basic edit can only be invoked by entering input from a cleared screen. Therefore, in this example, the terminal operator enters:

   ```
   LTISC1 | TRANX | Data...
   ```

   LTISC1 is the IMSA logical terminal name associated with an ISC session between IMSA and IMSB.
2. Basic edit edits the input data stream from the terminal, and the message is placed on the IMSA message queue with a destination of LTISC1.

3. On output to LTISC1, ISC support in IMSA:
   a. Strips the destination LTERM name (LTISC1) from the data stream
   b. Builds the FMH required to send the transaction to IMSB
4. The data stream that is sent to IMSB looks like:

```
FMH: DPN=SCHEDULER
```

```
FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T | TRANX | Data...
```

   a. DPN=ISCE because IMS ISC support supplies this value as a default if no DPN is supplied when output is to be sent to another subsystem.
   b. PRN= is not supplied and also not required for this example. TRANX is part of the user data. Only through the use of MFS can PRN= be supplied.
   c. RDPN= is not supplied and also not required for this example. Only through the use of MFS can RDPN= be supplied.
   d. RPRN=T is automatically inserted by IMSA as a default function of the message switching logic incorporated in IMS ISC.
5. The TRANX data stream is edited by ISC edit (ISCE) on input to IMSB, because the FMH specified DPN=ISCE. After editing, the data stream is placed on the message queues, and looks like:

```
TRANX | Data...
```

6. When scheduled, an MPP retrieves TRANX from the message queues and processes the transaction. Output inserted (ISRT) by the MPP to the originating input terminal looks like:

```
Data...
```

   This output can be inserted (ISRT) by the MPP to its I/O PCB (on the same parallel session) or to an alternate PCB. The I/O PCB or an alternate response PCB must be used if TRANX is a response-mode transaction. An alternate PCB could be used if TRANX were a nonresponse transaction. Alternate PCB output is sent on the same or a different parallel session, depending on the session to which the output LTERM is assigned.
7. On output from IMSB to IMSA, the FMH is built and sent with the data:

```
FMH:  DPN=SCHEDULER
```

```
FMH:  DPN=ISCE,PRN=T,RDPN=,RPRN | Data...
```

   The output is sent on the same session as that on which the input was received.
   a. DPN=ISCE is specified, because ISC support supplies this value as a default if no DPN is available from input, or through MFS when asynchronous output is to be sent to another system.
   b. PRN=T is supplied, because the input FMH to IMSB specified RPRN=T. IMS ISC support automatically wraps an input FMH RPRN value to the output FMH PRN field.
   c. RDPN= is not required but can be added by MFS DPM. The RPRN is not supplied, because the reply is returned on the same session as the input transaction. However, MFS can also be used to set the RPRN.
8. On input to IMSA, the reply from IMSB is edited by ISC edit (ISCE), because the FMH specified DPN=ISCE.

   Because a PRN is supplied in an input FMH, ISC edit uses the PRN as the IMS destination and appends the PRN value to the input data. After ISC edit, the message is placed on the input message queues and looks like:

```
T | Data...
```

If MFS had been used to process the reply within IMSA, the insertion of the LTERM name can be suppressed. See Example 6.

9. An IMS default MOD is used to format the data for output to the terminal. The data displayed to terminal T is:

```
T | Data...
```

Example 2. IMS-to-IMS application routing

The following figure illustrates IMS-to-IMS routing using an application program in both IMS subsystems.

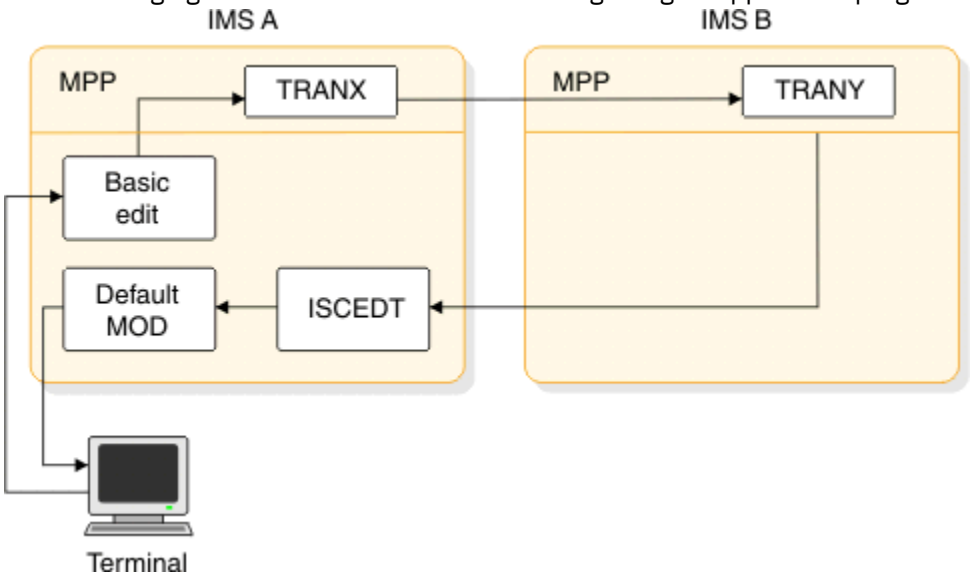


Figure 62. ISC example for IMS-to-IMS application routing

The input format from the terminal operator differs from that in Example 1 in that the routing can be supplied by TRANX in IMSA rather than by the terminal operator. Processing in IMSB and on the session is the same as that described for Example 1. Also see Examples 6 and 7 later in this topic to understand the interaction of MFS in the routing scenario.

Example 3. IMS-to-other-subsystem message switch routing

The function to be accomplished with the example shown in the following figure is the same as that in Example 1. However, the backend subsystem is either CICS or a user-written subsystem.

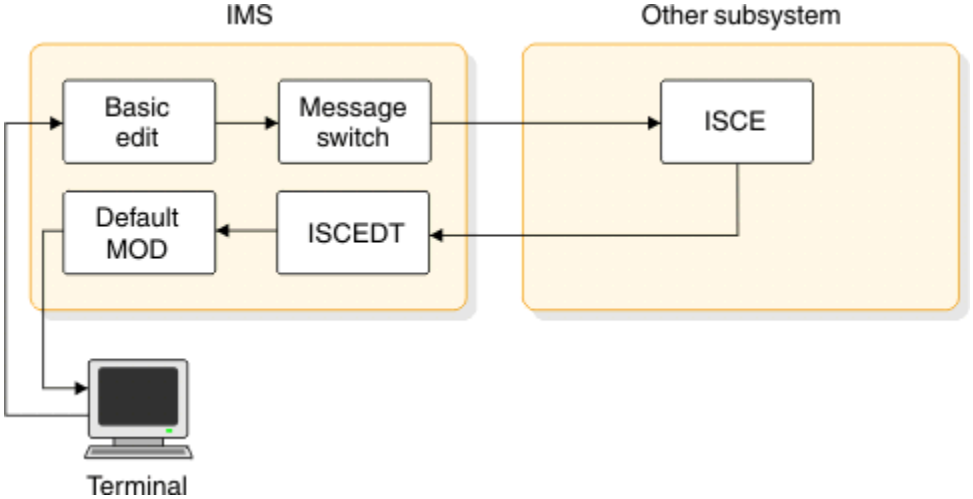


Figure 63. ISC example for IMS-to-other-subsystem message switch routing

The description of the activities taking place in IMS for this example parallels that in Example 1. The "other" subsystem is of interest and is described below.

1. Assume the other subsystem is CICS. Remember, from Example 1, the data stream sent from IMS looks like this:

   ```
   FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T | Data...
   ```

   The data is formatted in a way that ISCE expects to receive it. The IMS terminal operator who is entering the transaction should understand what that format is.
2. CICS must have a user-written transaction defined with transaction code ISCE, because the DPN represents a transaction code to CICS. CICS, after receiving the data stream described in item 1, invokes a transaction named ISCE, which must be specifically written for the ISC environment. This transaction examines the FMH values supplied and must use the RETRIEVE interface to obtain the user data and process TRANX.

   Because CICS supports transaction codes having a maximum of 4 characters, ISC edit has been given the alias ISCE.
3. The output from the transaction in CICS must look like the output that IMSB sent back to IMSA in Example 1. That is:

   ```
   FMH: DPN=ISCE,PRN=T,RDPN=,RPRN | Data...
   ```

   The CICS transaction ISCE actually gives CICS the format to use to build the output FMH (in contrast to Example 1, wherein IMS ISC support builds the FMH). In this example, the CICS transaction must supply DPN=ISCE and PRN=T to CICS to enable the proper FMH to be built.
4. When IMS receives this output data stream from CICS, all activities parallel those in Example 1.

A user can implement the functions of this example in a user-written subsystem by understanding the previous explanation of CICS functions and relating it to the functions implemented in the system.

Example 4. IMS-terminal-to-IMS-terminal message switch routing

The following figure shows a terminal T connected to IMSA effecting a message switch to terminal T2, which is connected to IMSB.

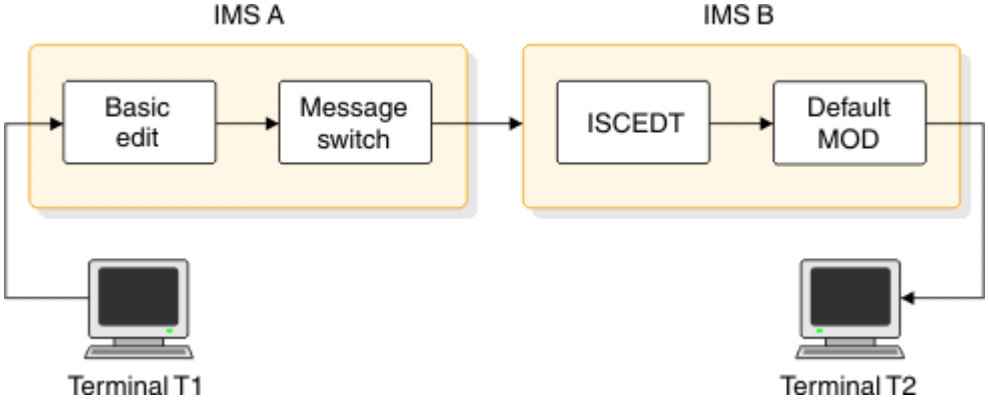


Figure 64. ISC example for IMS-terminal-to-IMS-terminal message switch routing

1. The terminal T enters the following data stream:

   ```
   LTISC1 | T2 | Data...
   ```

   LTISC1 is an LTERM name that IMSA has associated with an ISC session to IMSB. The terminal operator (T) must know this ISC LTERM name as well as the LTERM name of the destination terminal (T2) attached to IMSB.
2. ISC message switch support removes the LTERM name (LTISC1) before sending the balance of the message on the ISC session to IMSB. Therefore, the data stream that is sent on the ISC session looks like:

   ```
   FMH: DPN=SCHEDULER
   ```

```
FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T | T2 | Data...
```

a. DPN=ISCE is specified, because IMS ISC support supplies this value as a default if IMS does not supply a DPN when output is sent to another subsystem.

b. PRN= is not required, because the destination, terminal T2, is part of the data stream, but could be supplied or modified by MFS.

c. RDPN= is not supplied and is also not required for this example. This value could have been supplied by MFS.

d. RPRN=T, as in Example 1, is automatically inserted by IMSA as a default function of the message switching capability of IMS ISC. Because no reply is returned, this parameter is discarded by IMSB. MFS can be used in IMSB to make this parameter available to the operator of terminal T2.

3. Before the data stream is placed on IMSB's message queues, it is edited by ISC edit. In IMSB, ISC edit examines the data (because no PRN is available in the input FMH) to determine the destination. T2 is found to be the destination and the input message is placed on the message queues with a destination T2.
4. On output from IMSB to terminal T2, the data stream now looks like this:

```
T2 | Data...
```

5. A default system MOD would be used for output if T2 were a required MFS device.

Example 5. IMS-terminal-to-other-terminal message switch routing

The following figure is similar to Example 4; however, the back-end subsystem is CICS or a user-written subsystem.

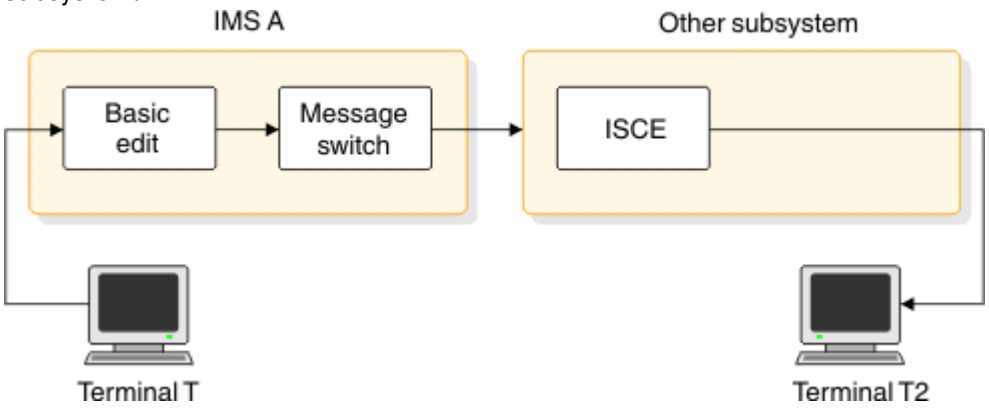


Figure 65. ISC example for IMS-terminal-to-other-terminal message switch routing

The description of the activities taking place in IMS in this example parallels the description in Example 1 exactly. The other subsystem is of interest is described.

1. Assume the other subsystem is CICS. Remember from Example 1 that the data stream sent from IMS looks like this:

```
FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T | Data...
```

2. As in Example 3, CICS must have a transaction code defined as ISCE.

   The data is formatted in the way in which ISCE expects to receive it. The IMS terminal operator who is entering the transaction should understand what that format is to be. CICS, upon receiving the data stream, must invoke a transaction named ISCE, specifically written for the ISC environment. This transaction uses the data in the input FMH and obtains the input data through the RETRIEVE interface. Because the PRN field in the input FMH is not initialized by IMSA to be equal to T2, the CICS application must initiate a message switch to terminal T2 from the transient data queue or start a new transaction naming T2 as the primary resource.

In this example, the output terminal T2 must be identified in the input data stream. If IMSA had initialized the PRN field in the input FMH to T2, CICS would have attached the transaction ISCE with T2 as the primary resource, and a SEND to terminal T2 could have been made directly.

3. In order for IMSA to set the PRN field, MFS would be required.

Example 6. IMS-to-IMS message switch routing with MFS

In the following figure, terminal T enters a transaction to be processed in the backend IMS subsystem.

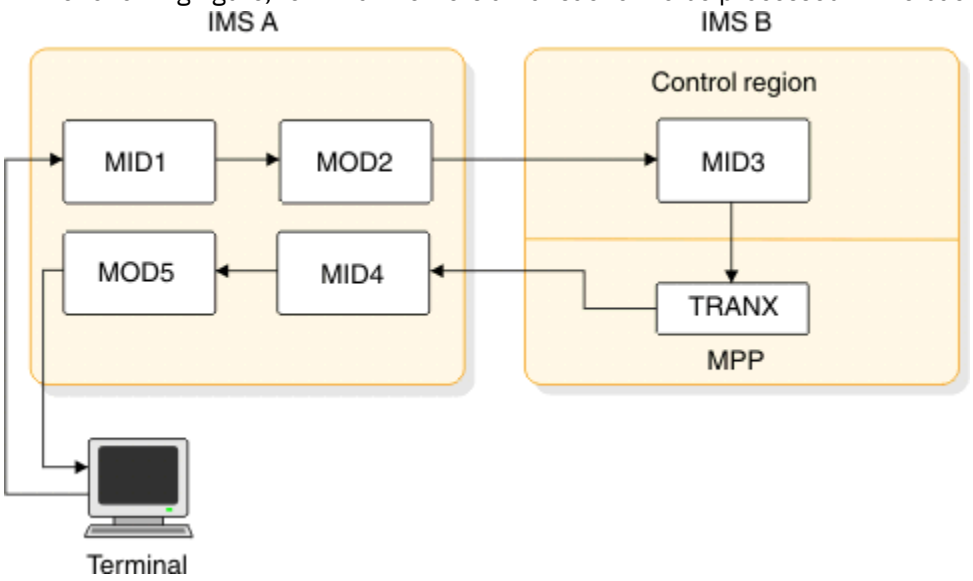


Figure 66. ISC example for IMS-to-IMS message switch routing with MFS

The reply from the back-end subsystem is to be routed back to the original terminal that entered the transaction (T).

1. Terminal T enters its input from a formatted screen. The terminal only needs to enter data.
2. After editing by MID1, the message is placed on the message queues and looks like this:

```
LTISC1 | TRANX | Data...
```

   The values LTISC1 and TRANX were appended to the data by MFS.

3. MOD2, to be used for editing the data stream that is sent to IMSB, is chained from MID1, and is chained to another MID (MID4) within IMSA. The data stream to be sent to IMSB looks like this:

```
FMH: DPN=SCHEDULER
```

```
FMH: DPN=MID3,PRN=,RDPN=MID4,RPRN=T
```

```
TRANX | Data...
```

   a. DPN=MID3 can be supplied by MOD2 or from a literal in the DOF associated with MOD2.
   b. PRN= is not needed, because TRANX is already in the data stream. It could have been supplied from MOD2 or its associated DOF.
   c. RDPN=MID4 is specified, because MID4 is chained to MOD2.
   d. RPRN=T is again automatically inserted by IMSA as a default function of message switching and not overridden by MFS.
   e. LTISC1 again has been stripped from the output message by the MFS MOD.

4. Upon receipt of the data stream from IMSA, IMSB edits the data using MID3, because the DPN in the FMH specified MID3.
5. The application receives and processes the following:

```
TRANX | Data...
```

6. Output from the application program is sent by IMSB to IMSA in the following format:

```
FMH: DPN=SCHEDULER
```

```
FMH: DPN=MID4,PRN=T,RDPN=,RPRN= | Data...
```

7. DPN=MID4 and PRN=T are automatically wrapped by IMS from the RDPN and RPRN values in the original FMH.
8. On receipt of this data stream, MID4 in IMSA edits the input and places the message on the queues for final output to terminal T in the following format:

```
Data...
```

   Because PRN=T is supplied, IMS uses it as the destination. Because MFS is also used to format the input reply, the PRN is not appended to the data.
9. Because MOD5 is chained to MID4, MOD5 is used to format the output to terminal T.

Several observations can be made about this example:

- MID3 in IMSB is not chained to a MOD. Therefore, the I/O PCB that the application program in IMSB sees does not contain a MODname to be used for output. If MID3 should have a chained MOD, the application ISRT call should have a blank MOD name to negate MFS editing on output.
- The application in IMSB has not changed the MOD name for output. If it does, additional MFS format design is required.
- The MPP in IMSB can be an already-existing program that also handles transactions from terminals connected to IMSB. The use of MFS can make the use of an ISC session transparent to the application.
- The original input terminal is not held in response mode.

Example 7. IMS-to-IMS application routing with MFS

The following figure can be understood in two contexts based upon an understanding of previous examples—the case shown in which ISC edit is used to edit the input in IMSB or the case in which MFS DPM-Bn is used to edit the input in IMSB.

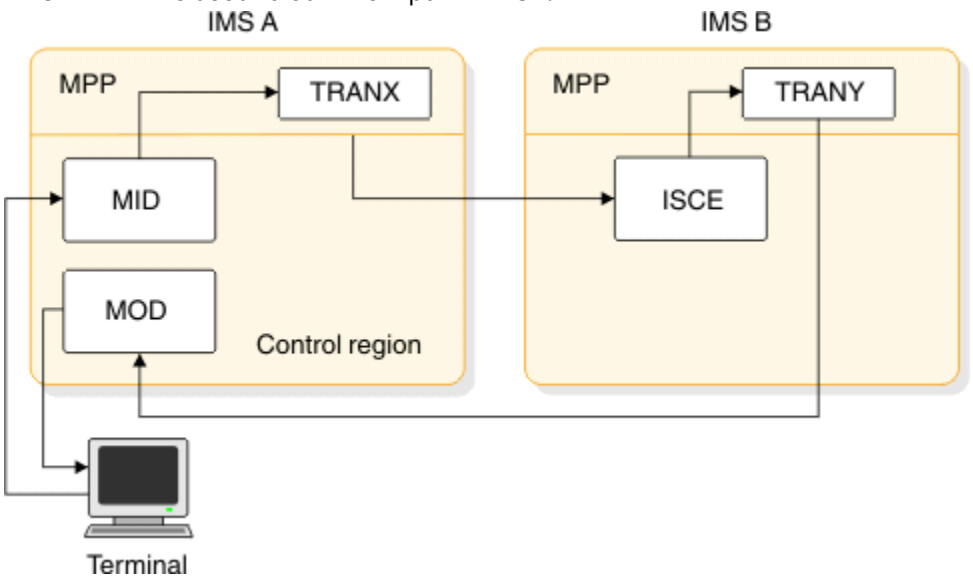


Figure 67. ISC example for IMS-to-IMS application routing with MFS

Example 7 emphasizes several points:

- Terminal T is completely handled by the front-end subsystem. Therefore, in this example, Terminal T can be held in response mode with IMSA.

- The data sent from IMSA is inserted (ISRT) by an alternate PCB and is asynchronous to the response mode transaction.
- IMSA could be connected to many IMS subsystems and, based on the input transaction, route the ISC traffic to the appropriate back-end IMSB.
- The MPP in IMSA can provide any required message routing information, such as selecting the appropriate back-end subsystem through an ISRT to an alternate PCB.
- MFS can be used in either or both subsystems to provide data stream formatting and routing.

Routing messages through MSC to an ISC LTERM

You can route messages across an MSC link to an ISC half session.

Suppose you have a setup similar that shown in the following figure. IMS A and IMS B connected by an MSC link. SYS C is connected to IMS B with an ISC link. If you want to route input messages and output replies between IMS A and SYS C, place the ISC routing parameters in the IMS prefix portion of the message. The MSC link-processing preserves these routing parameters.

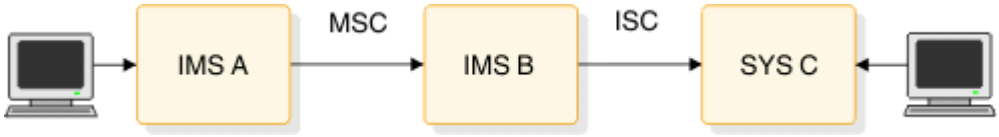


Figure 68. Routing of MSC to an ISC LTERM

Considerations for IMS-to-IMS ISC sessions

Consider the following information before configuring an ISC session between two IMS subsystems using static terminals.

Coexistence of ISC and MSC VTAM within one IMS subsystem

During IMS system definition, the TERMINAL, the MSPLINK macro statement, a dynamic CREATE MSPLINK command, or an ETO logon descriptor of one IMS subsystem must be defined with the same name as that defined on the COMM (ACB name) macro of the other IMS system.

IMS system definition allows both the dynamic CREATE MSPLINK command (or stage-1 system definition MSPLINK macro) and TERMINAL macros or an ETO logon descriptor to indicate the same remote subsystem name. However, the session qualifier information (the name used on the SUBPOOL macro and the partner-id parameter on the dynamic CREATE MSPLINK command or stage-1 system definition MSPLINK macro) must be unique. This allows MSC and ISC sessions to be defined and simultaneously active between two IMS subsystems. The number of ISC parallel sessions that can be active simultaneously between two IMS subsystems is the smaller of the values defined for the SESSION= parameter of the TERMINAL macro statements of the two IMS subsystems. (The SESSION= parameter is for statically defined terminals only.) The number of MSC parallel sessions that can be active simultaneously between two IMS subsystems is the smaller of the values defined for the SESSION= parameter of the dynamic CREATE MSPLINK commands (or stage-1 system definition MSPLINK macros) of the two IMS subsystems. The total number of MSC and ISC parallel sessions that can be simultaneously active is the sum of these two smaller numbers.

Defining single and parallel sessions

During IMS system definition, the TERMINAL macro statements or the ETO logon descriptors defining the ISC session(s) between the IMS subsystems must be defined identically as either single session or parallel sessions. However, when both subsystems are defined for parallel sessions, the number of concurrently active sessions or number of SUBPOOL macro statements need not be identical.

Without ETO, you can define up to 4,095 parallel sessions. With ETO, the number of ISC parallel sessions is limited primarily by the processor capacity and the amount of virtual storage above and below the line available to the control region.

Ensuring compatible buffer sizes

The VTAM output buffer size defined on the TERMINAL macro statement or an ETO logon descriptor of one IMS subsystem must be compatible with the "receive-any" buffer size defined on the COMM macro statement of the other subsystem.

Remote control of IMS through ISC

The IMS ISC message switch support or user-written applications allow a terminal operator within one IMS subsystem to effectively control the operation of another IMS subsystem. The IMS automated operator interface (AOI) facility can also be used to aid in the control of either IMS subsystem.

For statically defined terminals, you can authorize commands or transactions to ISC sessions within the remote IMS subsystem; all terminal operators and application programs within the local IMS subsystem that are authorized to send data on the ISC session can then access the remote subsystem. Therefore, use password security to restrict remote subsystem access to only specific authorized individuals and applications as a master terminal within the IMS subsystem.

You can provide additional security by specifying that all commands on the ISC session be issued between automated operator interfaces in both subsystems or issued to a single automated operator interface in the back-end subsystem.

A logical unit type 6.1 cannot be defined as the IMS master terminal during IMS system definition, nor can it be assigned as a master terminal using the IMS `/ASSIGN` command.

Restriction on IMS-to-IMS conversation mode

Conversation mode between two IMS subsystems is not supported. This is because conversations between IMS subsystems that are connected using ISC produce unpredictable session protocols and conversational transaction input when only one half session is in conversation mode. Further, the conversation terminates if both half sessions are in conversation mode.

Routing transactions to the back-end IMS

IMS ISC support allows the following to route transactions to a back-end IMS subsystem:

- Terminal operators using ISC message switches.
- Application programs using alternate PCB inserts within the front-end IMS subsystem.

In either case, the default action by the ISC support is to route a resulting transaction reply, in the form of a message switch, back to the originating terminal operator or transaction. Under these conditions, the transactions accessed within the back-end IMS subsystem should generally be defined as recoverable, because the response required for the message switch reply is also recoverable.

Irrecoverable transactions can be accessed, but require the use of MFS DPM-Bn to change the default destination set in the ISC message routing parameters from the LTERM associated with the originating terminal operator to an irrecoverable transaction within the front-end IMS. This avoids the protocol errors that occur when sending irrecoverable reply messages. The destination can be changed by MFS DPM either within the front-end IMS to modify or delete the default RPRN parameter sent on the transaction to the back-end IMS, or within the back-end IMS to modify or delete the wrapped PRN parameter sent on the reply to the front-end IMS. This irrecoverable transaction can then route the reply to the appropriate terminal operator by inserting to a modifiable alternate PCB.

Special support exists for front-end/back-end system utilization provided by the Front-End Switch exit routine.

Sending messages between the subsystems

Communication from an IMS front-end is always asynchronous. Therefore, messages are required to be sent and received with both the ATTACH and SCHEDULER FM headers. The exception to this is that system messages are sent with the ATTACH FM header only.

Protocol restrictions on IMS-to-IMS sessions

When an ISC session is between two IMS subsystems, certain types of protocols are not sent between the subsystems. This subtopic provides these protocols. If you are designing IMS-to-IMS sessions, you can skip the information on these protocols and headers provided in Chapter 30, “ISC protocols for VTAM connections,” on page 487.

- The data flow control commands BID, RTR, and RSHUT are not sent between the two IMS subsystems.
- The reset-attached process (RAP) FM header is not sent between IMS subsystems.
- The queue model (QMODEL) headers are not sent between IMS subsystems. This precludes IMS output demand-paging and all associated input QMODEL paging requests. Also, the ATTDQN parameter on the ATTACH FM header and the SCDDQN parameter on the SCHEDULER FM header are not supported in IMS-to-IMS sessions.
- The following error codes are not sent between two IMS subsystems:
  - LUSTATUS commit and function abort X'0864' and X'0866'
  - All sender ERP sense codes except X'0813' and X'0846'
  - Selective receiver ERP sense codes, except as listed under “Selective receiver ERP” on page 531.

Related reference

“System message process (SYSMSG) and related FM headers” on page 545
The system message process is indicated by a process name in the ATTACH FM header.

CREATE MSPLINK command (Commands)
Front-End Switch exit routine (DFSFEBJ0) (Exit Routines)

Statically defining an ISC node to IMS

A subsystem that is statically defined at system definition to IMS as an ISC node appears to IMS to be a terminal and to an IMS application program to be one or more logical terminals (LTERMs). Thus, an ISC node is defined using those system definition macros applicable to defining other VTAM terminals.

The following table shows the macro statements that define an ISC node.

Table 81. Macro statements for defining an ISC node

Macro statement	Use
COMM macro	Names the IMS subsystem to VTAM; names the ISC edit process and defines the receive-any (RECANY) buffer size.
TYPE macro	Identifies the unit type as an ISC node (UNITYPE=LUTYPE6).
TERMINAL macro	Identifies the other ISC node to IMS. Also provides sizes for the output and Fast Path buffers, limits for the input segment size, and processing options. Defines the number of sessions possible between two ISC nodes. Describes component characteristics.
VTAMPOOL macro	Begins the definition of the ISC LTERM users (subpools) used with parallel sessions.
SUBPOOL macro	Defines a set of LTERMs within the VTAMPOOL.
NAME macro	Names the LTERMs associated with a given terminal for a single session or a given user (subpool) for parallel sessions.

This topic more fully describes the macros that are key to defining an ISC session between two systems and some of the implications of selecting certain system definition options instead of others.

The following figure shows two IMS systems, NYIMS in a New York data center and SFIMS in a San Francisco data center. NYIMS and SFIMS are each defined to the other as an ISC node. The following paragraphs details the way in which these IMS systems would be defined during IMS system definition.

For ease of understanding, this description focuses on defining the IMS SFIMS to the IMS NYIMS. A similar system definition would be required to define IMS NYIMS to IMS SFIMS. Only those parameters of interest on the COMM, TERMINAL, and NAME macro statements are described.

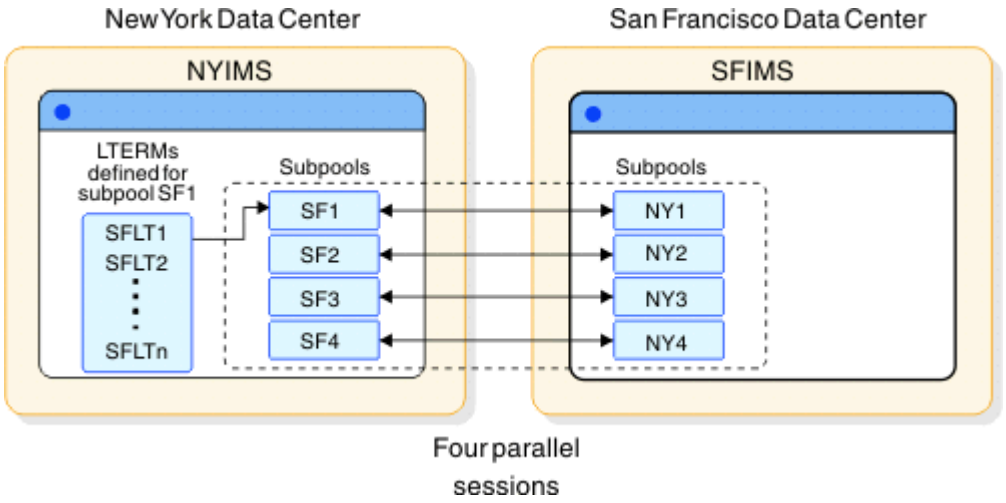


Figure 69. Two IMS systems defined to each other as ISC nodes

The parameter APPLID=NYIMS on the NYIMS system definition COMM macro makes that subsystem known by the node name NYIMS to VTAM, to itself, and to any other subsystems that need to access it. A similar system definition must occur for the San Francisco IMS SFIMS.

NAME=*nodename* on the TERMINAL macro statement of the NYIMS system definition is coded to show the node name of the SFIMS system (NAME=SFIMS). If SFIMS is an XRF complex, the node name should be the USERVAR or MNPS ACB associated with the SFIMS system.

The SESSION= keyword is related to parallel sessions rather than to a single session. (The SESSION= keyword applies to statically defined terminals only. It does not apply to the ETO environment.) SESSION= specifies the maximum number of parallel sessions that are allowed from NYIMS to SFIMS. It is possible to code one "parallel" session. A single session and one parallel session are not the same.

By coding SESSION= as greater than one, the NYIMS might maintain multiple concurrent sessions with SFIMS, up to the number specified by the SESSION= parameter. In the figure above, four paths are drawn between the two IMS systems. Each path represents a parallel session. To allow four parallel sessions in the NYIMS, code SESSION=4.

A major advantage of parallel session support is the ability to dynamically allocate LTERMs to that session based on the IMS /OPNDST command parameters.

Each parallel session is required to be identified to NYIMS using the SUBPOOL macro statement. In the example, to allow four concurrently active parallel sessions, four users (subpools) must be defined as follows in the NYIMS system definition:

```
VTAMPOOL
    SUBPOOL   NAME=SF1
    SUBPOOL   NAME=SF2
    SUBPOOL   NAME=SF3
    SUBPOOL   NAME=SF4
```

The VTAMPOOL macro statement begins the definition of the ISC subpools and must be coded if parallel sessions are to be used. The VTAMPOOL macro statement has no operands.

In Figure 69 on page 481 SFLT1, SFLT2, SFLT3,...SFLTn represent LTERM names that have been defined to the first SUBPOOL, SF1. The definition of LTERMs within a given subpool is accomplished by coding the NAME macro. Defining multiple LTERMs to a subpool provides several advantages. The first is that an LTERM or LTERMs defined to a given subpool can be reassigned (using the /ASSIGN command) to any other subpool. This capability gives flexibility in the operation of an ISC network.

The second advantage is with the COMPTn= keyword on the TERMINAL macro statement. This keyword can be used to define or assign certain characteristics to a given LTERM. COMPT= and ICOMPT= are the keywords on the NAME macro statement and are used to associate a component (COMPTn) with an LTERM.

The definition of SFIMS to NYIMS now looks like the following code:

```
TYPE     UNITYPE=LUTYPE6
TERMINAL NAME=SFIMS
         SESSION=4

VTAMPOOL
    SUBPOOL NAME=SF1
        NAME SF1LT1
        NAME SF1LT2
        ⋮
        NAME SF1LTN
    SUBPOOL NAME=SF2
        ⋮
    SUBPOOL NAME=SF3
        ⋮
    SUBPOOL NAME=SF4
        ⋮
```

Related concepts

“LTERM users (subpools) and components” on page 463

IMS user blocks are sets of IMS logical terminals (LTERMs) defined by the SUBPOOL macro during IMS system definition or dynamically created from ETO user descriptors.

Related reference

Macros used in IMS environments (System Definition)

Choosing parameters: system design considerations

The following topics further describe the parameters of the COMM, TERMINAL, NAME, and SUBPOOL macro statements.

COMM macro statement

The key parameters on the COMM macro statement are APPLID, RECANY, and EDTNAME.

The APPLID parameter defines the VTAM ACB name for IMS. The name specified for APPLID= must be the same as the ACBNAME= parameter of the VTAM APPL definition statement. If the ACBNAME= parameter is not specified, the APPLID= name must be the same as the name of the APPL definition statement that is used when defining IMS to VTAM. If the other subsystem is also an IMS, the name specified for APPLID= must be the same as that supplied on the NAME= keyword parameter of that IMS TERMINAL macro statement.

If you choose not to define the APPLID parameter on the COMM macro statement and instead use the job step name of the EXEC statement when initiating the IMS control region, VTAM attempts to match that name with either the ACBNAME= parameter or the name of the VTAM APPL definition statement, as described in the preceding paragraph.

When specifying the receive-any (RECANY) buffer size on the COMM macro, remember that a 28-byte overhead applies to the size of the receive buffers for ISC sessions. Therefore, the RECANY buffer size specified on the NYIMS system definition must be at least 28 bytes larger than the size specified for OUTBUF= on the TERMINAL macro of the SFIMS system definition for NYIMS. When defining SFIMS, the same considerations apply.

Other factors affect the size of the RECANY buffers. Because the specified RECANY buffer size applies to all VTAM devices, the size specified for RECANY might be larger than the size required for ISC due to the needs of other VTAM devices. Also, the FM header size must be added to the maximum data record size for both systems to determine the maximum ISC RECANY size.

The EDTNAME parameter specifies the alias that can be used for ISCEDT in your IMS system. In this system definition, the default name, ISCEDT, is used, making the specification of the EDTNAME= parameter unnecessary.

Related reference

COMM macro (System Definition)

NAME macro

The key system definition options for defining an ISC session on the NAME macro are: COMPT, OUTPUT, EDIT, and ICOMPT.

COMPT= and ICOMPT= specify the output and input components respectively to be associated with the LTERM being defined. The system definition example uses components defined as COMPT=1 and COMPT=2.

OUTPUT= should not be used on an ISC session.

Recommendation: If you do not want translation of ISC output to the session, specify ULC. If session output can be uppercase or lowercase data, or binary data, use ULC.

SUBPOOL macro

The SUBPOOL macro identifies ISC subpools to a system.

The key system definition options on the SUBPOOL macro are as follows:

- NAME
- MSGDEL

The NAME parameter identifies the ISC subpool.

In this example, the MSGDEL specification is the default, SYSINF0. When MSGDEL is specified on the SUBPOOL macro, it must match the MSGDEL specification on the TERMINAL macro.

Related concepts

“LTERM users (subpools) and components” on page 463

IMS user blocks are sets of IMS logical terminals (LTERMs) defined by the SUBPOOL macro during IMS system definition or dynamically created from ETO user descriptors.

TERMINAL macro

Several system definition keyword parameters on the TERMINAL macro are principal for defining an ISC session.

The principal system definition keyword parameters for defining an ISC session on the TERMINAL macro are as follows:

- OUTBUF
- MSGDEL
- COMPT1
- COMPT2
- COMPT3
- COMPT4

Additional parameters you must specify include:

- SINGLE or MULT
- VLVB or DPM-B1...DPM-B15

Specifying the OUTBUF= keyword parameter

IMS does not negotiate buffer sizes when initiating an ISC session, nor does IMS support RU truncation. When sending a negotiated bind, IMS sets its send (OUTBUF=) and receive (RECANY=) sizes in the bind.

When receiving the bind reply, IMS verifies that:

- Its send (OUTBUF=) size has not been reduced by the other subsystem.
- Its receive (RECANY=) size is not exceeded by the other subsystem's send (OUTBUF=) size.

When receiving a negotiated bind, IMS:

- Verifies that the send (OUTBUF=) size of the other subsystem does not exceed IMS's receive (RECANY=) size.
- Inserts the send (OUTBUF=) size into the bind reply.

The specified OUTBUF size must be large enough to include the largest output segment size plus the overhead for message headers. When defining the SFIMS system, the same considerations apply.

Specifying the MSGDEL= keyword parameter

A description of the ramifications of choosing SYSINFO or NONIOPCB is found in "LTERM users (subpools) and components" on page 463. In the example in that topic, the default for SYSINFO is used for both the TERMINAL and the SUBPOOL macros.

Specifying the COMPTn= keyword parameter

Up to four components can be defined for a given ISC node on the TERMINAL macro by using the keyword parameters COMPT1= through COMPT4=. COMPTn= allows flexibility in assigning characteristics to LTERMs. In the system definition example, component 1 is specified as having the characteristics SINGLE1, DPM-B1, and IGNORE. Component 2 is specified as having the characteristics SINGLE2, DPM-B2, and IGNORE.

Specifying the SINGLE or MULT parameter

The characteristics of SINGLE1, SINGLE2, MULT1, and MULT2 are described in "Determining output protocols" on page 467.

Specifying the VLVB or DPM-B1...DPM-B15 parameter

Selecting VLVB or DPM-Bn determines whether the component can use the Distributed Presentation Management feature of MFS. Although MFS is specified for a component by specifying DPM-Bn, its use is optional, on a message-by-message basis. DPM-Bn is used for this system definition example.

The selection of VLVB precludes the use of MFS-DPM for a component and indicates that variable-length, variable-blocked format is to be used instead of MFS for both input and output.

Specifying the IGNORE or 1, 2,...10 parameter

This parameter is used to specify a user-defined feature code for MFS DPM-Bn. The designated feature is used to select an MFS format (DPM-B1...DPM-B15) that contains the matching feature specification. IGNORE is used to specify that an MFS format (DPM-Bn) with the FEAT=IGNORE of the DEV statement is to be selected. IGNORE has been used for this system definition example.

Related reference

"COMM macro statement" on page 482

The key parameters on the COMM macro statement are APPLID, RECANY, and EDTNAME.

System definition summary

SFIMS from the previous example can be defined to NYIMS as shown in the following example.

```
TYPE     UNITYPE=LUTYPE6
TERMINAL NAME=SFIMS
     COMPT1=(SINGLE1, DPM-B1, IGNORE)
     COMPT2=(SINGLE2, DPM-B2, IGNORE)
     SESSION=4
VTAMPOOL
SUBPOOL NAME=SF1
   NAME SF1LT1, COMPT=1
SUBPOOL NAME=SF2
   NAME SF2LT1, COMPT=2
SUBPOOL NAME=SF3
   NAME SF3LT1, COMPT=2
SUBPOOL NAME=SF4
   NAME SF4LT1, COMPT=2
```

This definition does not correspond exactly to the figure in "Statically defining an ISC node to IMS" on page 480. In that figure, the SUBPOOL SF1 contains multiple LTERMs (to indicate that multiple LTERMs can be assigned to a SUBPOOL). Based on SINGLE1, SINGLE2, MULT1, and MULT2, the definition can now be simplified to show just one LTERM being defined per SUBPOOL.

The first SUBPOOL, SF1, has LTERM SF1LT1 defined with COMPT=SINGLE1. For asynchronous output from NYIMS to SFIMS, LTERM SF1LT1 is used as the destination LTERM for all:

- Alternate PCB output from application programs in NYIMS that generates asynchronous transactions in SFIMS
- Message switches originated by a terminal connected to NYIMS
- Replies to nonresponse transactions received by NYIMS from SFIMS

The characteristics of SINGLE1 (BB/message/EB) should allow one parallel session to handle all asynchronous input and output between NYIMS and SFIMS.

Three SUBPOOLS (SF2, SF3, SF4) have been defined with single LTERMs, each defined with a COMPT defined as SINGLE2. These three SUBPOOLS (sessions) can be used to send response mode transactions to SFIMS. Because of the nature of response mode transactions (session tied up from transaction origination until receipt of reply), three parallel sessions are defined to allow the user to minimize response mode bottlenecks between NYIMS and SFIMS.

In most IMS ISC definitions, the assignment of one LTERM to each SUBPOOL is sufficient to handle the communications traffic between the two nodes.

Not shown, but certainly to be considered, is the possibility of defining a fifth SUBPOOL within NYIMS to handle incoming SINGLE1 traffic from SFIMS. By design then, NYIMS could receive all SINGLE1 input from SFIMS on the session defined by the fifth SUBPOOL and send all SINGLE1 output on the session defined by the first SUBPOOL (SF1). This plan would result in minimal contention for asynchronous output between the two systems.

Related concepts

Defining VTAM terminals (System Definition)

Chapter 30. ISC protocols for VTAM connections

IMS uses ISC protocols to control sessions, data flow, and message routing over ISC VTAM connections. The following topics include the specific protocol information that you need to send and receive data with an ISC link.

Restriction: The ISC protocols for ISC VTAM connections do not apply to ISC TCP/IP connections.

Operating the network

Because ISC permits multiple sessions between logical units, the balance of this explanation of ISC differentiates between the terms logical unit, session, and half session.

Definitions:

- The term *logical unit* is used when referring to characteristics common to all sessions between two session partners or when it is not necessary to differentiate between individual sessions.
- The term *session* is used when describing a characteristic unique to a given connection (session instance) between logical units.
- A *half session* describes characteristics unique to one of the session partners.

Making IMS ready

Use the IMS /START command with the DC keyword to make IMS ready to receive VTAM logon or BIND SCIP (session initialization) requests.

The DC keyword initiates IMS data communications processing, opens the VTAM access method control block (ACB) if it is not already open, and enables the IMS VTAM logon exit. Any logon requests received by VTAM before the IMS /START DC command is issued but after the ACB has been opened are queued in VTAM until the /START DC command is completed. If VTAM is active when IMS is initialized, and the DFSDCxxx PROCLIB member keyword VACBOPN=INIT, then the IMS VTAM ACB is opened. If DFSDCxxx PROCLIB member keyword VACBOPN=DELAY, then the IMS VTAM ACB open is delayed until the /START DC command is processed.

The /START DC command also tells VTAM to pass any queued VTAM logon requests to IMS.

Starting an IMS network for ISC

Before any sessions can be established, VTAM and NCP must be active. In addition, all logical units must be online (activated by the VARY command).

You do not need to perform message resynchronization after a cold start.

Related concepts

Starting Transaction Manager (Operations and Automation)

Related tasks

“Controlling the session (session control protocols)” on page 488

Session initiation includes initiating a session instance, binding the session, and, if necessary, ensuring that the half sessions are in sync.

Shutting down an IMS network for ISC

Use the IMS **/CHECKPOINT** command to terminate the network and shut down IMS.

The format of the **/CHECKPOINT** command determines whether the network termination occurs immediately or waits for processing to complete:

- **/CHECKPOINT FREEZE|DUMPQ|PURGE** immediately terminates the session for all logical units, as follows:

© Copyright IBM Corp. 1974, 2020

FREEZE
Immediately after current input/output message

DUMPQ
After control blocks are past checkpoints

PURGE
After all queues are empty

- **/CHECKPOINT FREEZE|DUMPQ|PURGE QUIESCE** allows all network nodes to complete normal processing before shutting down.

The IMS command **/STOP DC** also shuts down an IMS network at completion of processing.

Also, certain VTAM commands, such as **VTAM HALT NET**, shut down the network.

Related tasks
Shutting down an IMS network (Operations and Automation)

Controlling the session (session control protocols)

Session initiation includes initiating a session instance, binding the session, and, if necessary, ensuring that the half sessions are in sync.

When both half sessions have agreed to the bind and are in sync, normal traffic flow is initiated using the VTAM start data traffic (SDT) command.

Initiating an ISC VTAM session

A session must be established before data can be transmitted between another logical unit type 6.1 and IMS. If message resynchronization is required, it is performed after the bind.

IMS can be requested to initiate a session in one of the following ways:

- An ISC logical unit requests session initiation by sending the "initiate-self" command. VTAM verifies the command and passes the request to IMS.
- An ISC logical unit sends BIND to initiate a session with IMS.
- The z/OS VTAM network operator requests session initiation on behalf of the logical unit by using the z/OS VARY command with the LOGON option. VTAM processes the request and passes it to IMS. The VARY command cannot be used to initiate parallel sessions.
- VTAM passes to IMS a logon or BIND SCIP request for each logical unit defined to VTAM as belonging to IMS. (This method cannot be used to initiate parallel sessions.)
- An authorized IMS terminal operator requests session initiation for an ISC logical unit by entering the IMS **/OPNDST** command.

IMS can initiate (send BIND) or accept (receive BIND) a session if all the following conditions are met:

- The IMS master terminal operator has issued a **/START DC** command.
- The logical unit name is known by IMS.
- The logical unit is not stopped (**/STOP** command) within IMS.
- The logical unit has not reached the maximum allowable sessions defined to IMS. (This applies to statically defined terminals only.)
- If the session is to be parallel, the CINIT or BIND session qualifier field must contain a valid LTERM subpool name that defines the message queue set to be used. A valid subpool name is one that is not stopped or allocated (except during session restart). The session qualifier fields must not be supplied when requesting session initiation with a logical unit defined to IMS as single session.
- For statically defined terminals, the MSGDEL option for the LTERM subpool as specified on the SUBPOOL macro does not conflict with the MSGDEL option specified for the session on the TERMINAL macro.
- No invalid or conflicting parameters are indicated on either a CINIT, BIND, or negotiable BIND reply.

Related concepts

"Statically defining an ISC node to IMS" on page 480
A subsystem that is statically defined at system definition to IMS as an ISC node appears to IMS to be a terminal and to an IMS application program to be one or more logical terminals (LTERMs). Thus, an ISC node is defined using those system definition macros applicable to defining other VTAM terminals.

Related reference

Bind parameters for SLU P and LU 6.1 (System Programming APIs)

Establishing a connection with an XRF complex

An XRF complex can be a session partner.

There are two types of XRF complexes: those that use a USERVAR to manage sessions during an XRF takeover and those that use an MNPS ACB to manage sessions during an XRF takeover.

In either case, if the active IMS subsystem in the XRF complex fails, the alternate IMS subsystem re-establishes the session during takeover. Until the ISC session between the old active IMS and the remote system terminates completely, session initiation fails. Upon takeover, the alternate IMS subsystem becomes the active subsystem and retries session initiation every 30 seconds (for a maximum of 20 attempts) until the session initiates successfully.

Connecting to an XRF complex that uses an MNPS ACB

To establish a connection with a session partner that is an XRF complex that uses an MNPS ACB, simply initiate the session with the XRF complex by specifying the MNPS ACB name as you would to connect to any other non-XRF IMS system.

Connecting to an XRF complex that uses a USERVAR

To establish a connection with a session partner that is an XRF complex that uses a USERVAR, the steps you take differ depending on whether the session is initiated from the XRF complex or the IMS system outside of the XRF complex.

- If the non-XRF half session initiates the session with IMS, the non-XRF half session should:
  1. Identify the APPLID of the currently active IMS system by issuing the following macro:

     ```
     INQUIRE OPTCD=USERVAR,AREA=area,AREALN=8
     ```

     If the VTAM network is Release 3.2 or later and USERVAR Management Enhancement is installed, you do not need to issue the INQUIRE macro.

     With this request, the AREA parameter specifies the address of the area with the USERVAR name for the active XRF system. The USERVAR is then replaced with the APPLID of the currently active IMS system.
  2. Issue a session-initiation request by specifying the APPLID of the active IMS system.

     If the VTAM network is Release 3.2 or later and USERVAR Management Enhancement is installed, issue a session-initiation request by specifying the USERVAR of the XRF system.
- If the IMS system in the XRF complex initiates the session, the IMS system:
  1. Issues the session-initiation request, adding the USERVAR assigned to the BIND in the user data field.
  2. Validates the session-initiation request by referencing the USERVAR in the user data field located at the end of the request. Initiation is not apparent to the user.

Binding the session

In an ISC session, IMS can assume either the primary half-session role (send the BIND) or the secondary half-session role (receive the BIND). However, when session restart and recovery are required, session polarity must be maintained.

Definition: *Session polarity* means that the same session role (primary versus secondary) is in effect at the point of failure and is reestablished by the session initiation request. Otherwise, the request is rejected. When the session is initiated using an IMS **/OPNDST** command and session restart and recovery are not required, IMS requests to be the primary half session.

Negotiable versus nonnegotiable BIND

As the primary half session, IMS sends either a negotiable or nonnegotiable BIND, depending on parameters in the VTAM mode table.

The mode table entry is indicated on the VTAM CINIT or IMS **/OPNDST** commands, or defined on the TERMINAL macro during IMS system definition.

Definitions:

- When sending a *negotiable BIND*, IMS sets the bind parameters. Because the secondary half session can change some parameters, all the parameters are checked for validity by IMS when a negotiable BIND response is received.
- Prior to sending a *nonnegotiable BIND*, IMS checks all parameters from the mode table entry for validity. The session is terminated if IMS finds any incompatible parameters.

As the secondary half session, IMS receives both the negotiable and the nonnegotiable form of BIND. When receiving a negotiable BIND, IMS checks the bind parameters, except the secondary network addressable unit (NAU) protocol field, for validity. IMS then sets the secondary NAU protocol field for the negotiable BIND response.

When receiving a nonnegotiable BIND, IMS checks the bind parameters, except for "STSN required" and "BIS sent," for validity prior to accepting the BIND. IMS must then operate under the secondary NAU protocol definition provided in the bind.

Related reference

IMS as primary half session (System Programming APIs)

Bind parameters for SLU P and LU 6.1 (System Programming APIs)

Binding single or parallel sessions

Requirements for binding a session differ for single and parallel sessions.

To bind a single session, either negotiable or Nonnegotiable BIND is sent. The ISC logical unit with which IMS is communicating must have been defined with a static set of LTERMs during IMS system definition. IMS ignores the session qualifier pair (SQP) field on both CINIT and BIND.

To bind a parallel session, the CINIT and BIND parameters must include a SQP field to identify the specific parallel half-session instance between IMS and a logical unit. (All sessions created dynamically using the ETO feature are parallel sessions.) This field in the bind parameters contains both the primary and the secondary session qualifiers. The half-session name of the ISC node communicating with IMS consists of the logical unit name concatenated with its associated session qualifier. The IMS half-session name is the IMS ACB name concatenated with the session qualifier associated with IMS. The IMS session qualifier is the subpool name. Both half-session names are saved across session and IMS subsystem failures and are used to allocate, or validity-check when warm starting, the LTERM subpool to be used for the session.

Resolving a bind race

A race occurs when IMS and another logical unit simultaneously send BIND requests to each other and the two half-session names are mirror images.

That is, the primary logical unit name and session qualifier concatenated with the secondary logical unit name and session qualifier of one side is equal to the primary logical unit name and session qualifier concatenated with the secondary logical unit name and session qualifier of the other side. The logical unit that wins and becomes primary is the one whose name (the primary logical unit name taken from each half-session name) is at the top of the standard collating sequence. This test prevents both sides from rejecting each other, resulting in no session.

After a successful session bind

Following a successful session bind, both half sessions must perform a resynchronization process. Message resynchronization is not performed when IMS is cold started or when a previous session instance between IMS and another logical unit has been normally shut down using the Stop Bracket Initiation/Bracket Initiation Stopped (SBI/BIS) procedure.

Related concepts

"Resynchronizing sessions" on page 491
To maintain the integrity of recoverable resources, messages, and queues in IMS across both subsystem and session failures, both half sessions must maintain the session information required for the resynchronization process.

Related tasks

"Terminating an ISC VTAM session" on page 499
Terminating an ISC VTAM session releases a logical unit from its current connection to the VTAM application program, making the LU available for sessions with other VTAM applications, or terminating communications altogether.

Related reference

"Symmetrical session shutdown for LU 6.1 (SBI and BIS)" on page 540
Two data flow control commands allow a symmetrical and orderly termination for peer level LU 6.1 half sessions: stop bracket initiation (SBI) and bracket initiation stopped (BIS).

Resynchronizing sessions

To maintain the integrity of recoverable resources, messages, and queues in IMS across both subsystem and session failures, both half sessions must maintain the session information required for the resynchronization process.

Message integrity cannot be maintained without user intervention when either or both subsystems incur an error or when a user restart procedure destroys this information.

Resynchronization is required when it is possible for a recoverable work unit to be indoubt on a flow (primary-to-secondary or secondary-to-primary).

Definition: A *work unit* includes all transmissions between sync points within a bracket, as illustrated in the following figure. A sync point might have been requested by one or both half sessions without having been acknowledged. These conditions can be caused by a subsystem or session failure or an abnormal completion of a shutdown sequence.

Performing message resynchronization is unnecessary following a normal shutdown sequence (unless nonnegotiated BIND was sent), because both half sessions can come to a controlled, mutual understanding that no additional normal message traffic or sync-point requests are to occur prior to session termination. Message resynchronization is always required following nonnegotiable BIND to allow error conditions detected by the secondary half session to be communicated to the primary half session.

The half-session pairs resynchronize with the VTAM BIND, set-and-test-sequence-numbers (STSN), and start data traffic (SDT) commands. The STSN command allows both half sessions to reestablish sync-point information (session sequence numbers), which is being maintained by both half sessions.

When message resynchronization is necessary, it must be completed successfully before either half session can resume normal data transmission.

The following figure shows two work unit examples.

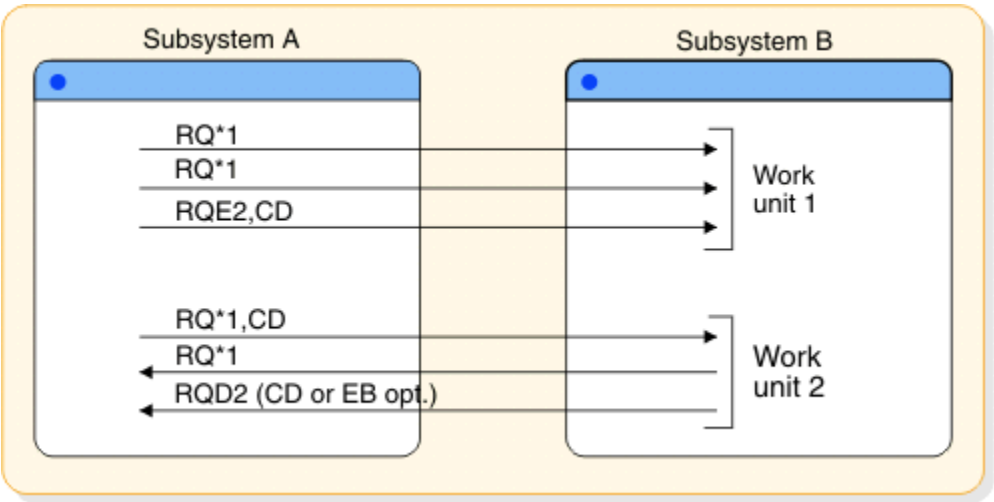


Figure 70. Work unit examples

In `work unit 1` of the preceding figure, a reply to the exception response request and CD creates an implied sync point. In `work unit 2`, sending the DR2 response to the RQD2 creates a sync point.

Related concepts

"Keeping half sessions synchronized" on page 506
Sync-point responses (DR2) are used between ISC session partners to ensure that both partners' sync-point managers can commit or back out recoverable resources synchronously.

Related reference

"Sync point and response requirements" on page 508
The IMS input/output message flow can be represented as an input/output flow from a sequential queue data set.

Designing restart resynchronization procedures

Resynchronization might or might not be required when IMS is restarted and when a session is established or reestablished. This topic addresses considerations for resynchronization at IMS restart and at session restart.

IMS restart provides for message recovery during:

- A normal IMS subsystem restart procedure, by restoring the IMS message queues and session restart information to the last or specified checkpoint
- An emergency IMS subsystem restart procedure by restoring the IMS message queues and session restart information to their states just prior to failure

If the failure occurred prior to a normal shutdown sequence between IMS and another logical unit, ensure that the session is restarted using the same half-session pairs (half-session names) that were in session at the time of failure. Further, when the session is resumed, the half-session pairs must maintain their previous relationship—that is, the former primary half session must again be primary, and the former secondary half session must be the current secondary.

Maintaining sequence numbers

To allow session recovery and message resynchronization across session and subsystem failures, both half sessions must maintain a checkpoint of sequence numbers and other indicators.

Specifically, a checkpoint must be maintained of the following:

- Three sequence numbers that are produced when the session was last active

  The three sequence numbers are the potential (pending) and committed sequence numbers for the flow sent by the half session and the last committed sequence number for the flow received by the half session.
- An indicator produced when a unilateral decision was made by a half session to back out or commit a work unit that was left pending during the session outage
- An indication of the direction of the decision

Sequence number mismatches and incorrect decisions to commit or back out a recoverable work unit are detected by comparing the sequence numbers sent or received on the VTAM set-and-test-sequence-number (STSN) command with the checkpointed resynchronization information. Each half session can detect invalid sequence number mismatches on their outbound flow and reject the resynchronization request. Each half session can also agree or disagree to unilateral decisions made by the other half session to commit or back out a work unit. This agreement or disagreement is based on the detection of a sequence number mismatch on the inbound flow and involves a second STSN when an incorrect decision is made on the STSN receiver's inbound flow. The STSN command receiver does this before responding to the second STSN command (which is sent in response to the receipt of TEST NEGATIVE on the first STSN). This second STSN informs the STSN receiver that a wrong decision had been made on its inbound flow.

Recovering sessions with cold start

Invalid sequence number mismatches and mismatches where no unilateral decisions have been made often indicate a subsystem restart from an incorrect log or incorrect checkpoint on the log. This situation can require forcing the recovery of the ISC session by cold starting one or both half sessions.

IMS allows authorized terminal operators to change IMS ISC session state from recoverable to cold start when necessary by using the IMS **/ASSIGN** (subpool to VTAMPOOL) command before attempting to initiate a session. When a session is changed to a cold-start state, the subpool associated with that session is made available for allocation to any ISC cold-startable parallel session with any node.

Session recovery where one or both half sessions are cold started is not considered a major error because, by definition, the cold starting half session has no information from the previous active session with which to negotiate (agree or disagree) recovery or resynchronization.

Controlling unilateral decisions about pending work units

To prevent resources from accidentally getting out of synchronization between subsystems, you can specify whether to continue session resynchronization after a session outage, where the other half session has made a unilateral decision to commit or back out a pending unit of work.

IMS allows system definition to allow resynchronization without regard to inbound sequence number mismatches (OPTIONS=FORCSESS on the TERMINAL macro statement or an ETO user descriptor), or to only allow resynchronization when the inbound sequence numbers agree (OPTIONS=SYNCSESS). A keyword on the IMS /CHANGE command allows an authorized terminal operator to override the system definition specification for a single attempt to initiate a session. The effects of the CHANGE command are reset to the original system definition specification after one session initiation attempt. IMS does allow unilateral decisions to back out during session outages. However, using the IMS /DEQUEUE command for a session (terminal, and optionally, the subpool parameter) or LTERM during a session outage where an output-message sync point is pending is considered to be a unilateral decision by an authorized terminal operator to commit the pending output.

A pending output sync-point response can be determined by using the IMS /DISPLAY command. When two IMS subsystems are connected by an ISC session, the FORCSESS option must be in effect on the opposite IMS subsystem from one where a pending output message was dequeued (committed) using a /DEQUEUE command, or session resynchronization fails with message DFS2065. Using FORCESS rather than SYNCSESS has no other effect between two IMS subsystems.

Recovering from in-bracket failures

When a session has failed while in-brackets, it might be necessary to restart the failed bracket when the session is restarted.

The bracket state manager and the half-duplex reset flags in the bind can be set in either the BIND request or the negotiable bind response by either half session. These flags are set to in-brackets/SEND or RECEIVE to indicate the possibility that the process previously attached must be restarted.

Response or conversational output available at restart

If, during session initiation, IMS has response mode or conversational output available to be sent or is in a sync-point response pending state, the bracket state manager reset and half-duplex flags within the IMS portion of the bind or bind response are set to in-brackets/SEND by IMS.

If IMS is in conversation mode but has no conversational output available or pending, these same bind or bind reply flags are set to in-brackets/RECEIVE, because input is required to continue the conversation. IMS does not accept a nonnegotiated BIND or negotiated BIND response indicating IMS to be in-brackets/RECEIVE when conversation or response mode output is available. Unless a sync point is pending from a previous session, the session is resumed and the STSN that is sent must receive a TEST POSITIVE reply to the SET AND TEST option in order for processing to continue.

When IMS is in conversation or response mode and receives a nonnegotiated BIND or a response to a negotiated BIND indicating between-brackets, IMS attempts to terminate the conversation or response mode. This termination is just like that which occurs if end-bracket is received on an FMH7 or LUSTATUS while IMS is in conversation or response mode during normal data flow active state (after SDT).

Conversation and response mode termination is only assured when the output reply message is available to be sent or the half session is waiting for the next conversational input. Therefore, IMS restricts attempts to terminate conversation or response mode by binding between-brackets when the output reply is available to be sent or no conversational input is required. If no output reply is available to be sent or no conversational input is required, a warning message is sent to the IMS master terminal operator with instructions to retry session initiation later.

If IMS receives a negotiated BIND indicating in-brackets/SEND and is not in conversation or response mode, a BIND response is sent indicating between-brackets. IMS sends LUSTATUS - NO-OP indicating end-bracket immediately after SDT when IMS is not in conversation or response mode and when receiving a non-negotiated BIND or negotiated BIND response indicating in-brackets/SEND. LUSTATUS - NO-OP is sent because transaction restart is not possible under these circumstances.

It is possible that IMS is left in-brackets/SEND after a BIND or BIND response, because a conversation or response mode output sync-point response is pending from the previous session. The output message might be dequeued if STSN processing indicates that the response was sent. In this case, an LUSTATUS - NO-OP indicating end-bracket is to be sent after SDT if the output message is a response mode reply. An LUSTATUS - NO-OP indicating change-direction is sent if the output message is a conversational reply, because input is required to continue the conversation.

If STSN indicates that the conversation or response mode message was not acknowledged, IMS again places the message in a sync-point response-pending state as if it had been sent with change-direction and as if it had requested an exception sync-point response. Any normal flow reply is then considered implicit acknowledgment of the pending output message and causes it to be dequeued by IMS even if the reply was an LUSTATUS - Abort or an ATTACH ATTDPN=SYSMSG. These replies indicate additional error conditions associated with the message's scheduling or execution. Had the session not failed, these error conditions would have been reflected by an exception response to the sync-point request rather than the LUSTATUS - Abort or ATTACH ATTDPN=SYSMSG. Depending on the sense code used, the exception response might have caused the message to be returned to the message queue (backed out)

for retransmission rather than dequeued (committed) as occurs for LUSTATUS - Abort or ATTACH ATTDPN=SYSMSG resulting when both a session and an application failure occur together.

Session failures without IMS failure

A session failure not also involving an IMS subsystem failure might occur before IMS returns a requested sync-point response for a response mode or conversational input message.

When the session is reestablished, the DFC state set by IMS using the bind is IMS in-brackets/SEND. The subsequent STSN sequence number recovery might reflect either that the input message has been committed (the transaction has reached a sync point after inserting a reply message) or that the input message has not yet been committed. In either case, IMS being bound as in-brackets/SEND indicates that the input message has been received and the other half session should wait for the reply message. Because session restart resets the original sync-point request, the reply message now becomes an implicit sync-point response to the original input message.

IMS recovers the fact that the session was in conversational mode (and also recovers associated input messages and output replies) across IMS outages. The fact that the session was in response mode (and any associated output replies) is only recovered across an IMS outage if the failure occurs after the transaction sync point that made the response mode reply available.

Session failures because of IMS failure

A session failure that also involves an IMS subsystem failure might occur before IMS responds to a synchronization request on response mode (if the message was recoverable) or conversational mode input.

When the session is reestablished, the DFC state set by IMS by the bind indicates IMS as bound in-brackets/SEND. In either case, because of the IMS restart process from the subsystem log, the subsequent STSN sequence number recovery reflects that the input message has been committed even if that commit has not yet actually occurred. However, this condition produces inconsistent results only if the transaction subsequently abnormally terminates after being restarted. In this case, the response mode output reply message is made available for asynchronous (ATTACH EB or ATTACH ATTDPN=SCHEDULER) delivery on the recovered ISC session.

Recoverability of commands and execution modes

IMS commands (except **/DIS**, **/RDIS**, and **/FOR**) and test-mode input are executed immediately without being placed on an input message queue. These commands complete all processing prior to causing output to be made available using a transaction sync point. Any failure causes the command or test mode transaction input and associated output message to be backed out and discarded.

IMS commands **/DIS**, **/RDIS**, and **/FOR** produce asynchronous queued output. Output that is enqueued prior to the failure is recovered and made available for asynchronous transmission when the session is reestablished.

Fast Path, recoverable response mode, and recoverable conversational mode transactions are backed out and discarded if the failure occurs before a transaction sync point. The fact that the session was in response mode is not recovered.

Coordinating the restart process

Half sessions use rules to coordinate restart after a session failure.

The rules used by the half sessions include:

- The restart always occurs from the most currently completed sync point.
- The session bind establishes the half-duplex state at the current sync point.
- When the primary half session wants to restart:
  - It sends a BIND request to select the proper half-duplex state.

  - The secondary half session cannot change this state in the BIND response, unless the secondary half session does not want to restart.
  - When the secondary half session does not want to restart, it sends a BIND response to preclude restart by setting the bracket state to between-brackets.
- When the primary half session does not want to restart:
  - The primary half session sends a BIND request setting the bracket state to between-brackets.
  - If the secondary half session wants to restart, it sends a BIND response that establishes the proper half-duplex state and sets the bracket state to in-brackets.
  - If the secondary half session does not want to restart, the secondary half session' response to the BIND does not change (it matches the state set by the primary half session).
- STSN is used to resynchronize any pending requests for sync-point responses.
  - If no mismatch occurred, agreement exists on the current sync point. The restart is attempted after sending start data traffic (SDT).
  - It might be necessary for the half session in send state to send LUSTATUS - NO-OP, CD to adjust the half-duplex state to the current sync point based on the STSN.
  - If a mismatch occurred and the session continues, the half session in send state sends LUSTATUS - NO-OP, EB to place the session into contention.
- The session in send state at the final restart point reestablishes the session by sending an explicit ATTACH.

Half sessions assuming a secondary role must reject a session bind in these situations:

- Session bind parameters indicate in-brackets, "other half session speaks first", and response mode or conversational output (or equivalent) are immediately available at data traffic active state. This is a session restart logic error.
- On restart following an abnormal failure or shutdown sequence, session bind parameters indicate half-session names different from those in effect during the previous session between IMS and another logical unit.

In an ISC session, either subsystem can assume a primary or a secondary half-session role, and either subsystem can request message resynchronization; therefore, a set of rules must be established by which these half-session pairs can maintain message or sync point integrity.

These topics describe the format of the STSN and how the STSN is used to complete resynchronization and recovery.

Related reference

“LUSTATUS protocol” on page 528

IMS sends and receives LUSTATUS as summarized in the following table. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this topic.

Determining session synchronism using STSN

The need to resynchronize can be communicated during bind negotiation; two flags in the BIND request are used to determine the requirement to resynchronize and to return the half sessions to the state that existed at the time of session termination.

If both half sessions have terminated normally (that is, neither had any outstanding traffic to handle), the session is restarted as though it were a new session—bind parameters are sent, and no requirement exists that the relative positions of the session partners be maintained, nor that the same half-session names be used. The two flags upon which this determination is made are:

- Sequence number indicator

  1 = Sequence numbers available
  0 = Sequence numbers not available

- Bracket initiation stopped (BIS) indicator

  1 = BIS sent
  0 = BIS not sent

The following table is a matrix describing the states set by these two flags in relation both to the bind sender (primary half session, or PHS) and the bind receiver (secondary half session, or SHS).

When both half sessions are in a "COLD START" state, no sequence numbers are available or required to be sent. Session shutdown is such that resynchronization is not required. The bind is negotiated and the session started in accordance with the bind parameters. For a nonnegotiable BIND, some information required by both half sessions is not available until STSN flows exist. Therefore, for a nonnegotiable BIND, STSN is always sent.

Table 82. BIND action/response matrix

	Bind receiver (SHS): Numbers not Available	**Bind receiver (SHS): Numbers available, BIS sent**	**Bind receiver (SHS): Numbers available, BIS not sent (See Table 83 on page 501)**
Bind sender (PHS): Numbers Not available	COLD STSN not required Send RESET to NOBB	COLD STSN not required Send RESET to NOBB	COLD/WARM mismatch STSN sent
Bind sender (PHS): Numbers available, BIS sent	COLD STSN not required Sent RESET to NOBB	COLD STSN not required Send RESET to NOBB	WARM STSN sent
Bind sender (PHS): Numbers available, BIS not sent (Table 84 on page 502)	WARM/COLD mismatch STSN sent	WARM STSN sent	WARM STSN sent

IMS sets the "BIS sent" and "sequence numbers available" flags in the negotiable BIND and BIND response (or PHS and SHS, respectively) as follows:

- Session cold start following IMS cold start or normal session termination:

  BIS not sent
  Sequence numbers not available
- Session restart
  - Previous session terminated normally following attempted normal termination by IMS:

    BIS sent
    Sequence numbers available
- Previous session terminated abnormally (normal termination by IMS not attempted):

  BIS not sent
  Sequence numbers available

Related reference

“Performing the resynchronization” on page 498
When a session is reestablished (or established for the first time), the state of each flow is represented by the sequence numbers that are available for synchronization in each half session.

“Symmetrical session shutdown for LU 6.1 (SBI and BIS)” on page 540

Two data flow control commands allow a symmetrical and orderly termination for peer level LU 6.1 half sessions: stop bracket initiation (SBI) and bracket initiation stopped (BIS).

"STSN command format" on page 503
Both the STSN command and STSN response contain a 5-byte data field.

Performing the resynchronization

When a session is reestablished (or established for the first time), the state of each flow is represented by the sequence numbers that are available for synchronization in each half session.

The sequence numbers available for synchronization in each half session can represent the following states for each flow:

- COLD

  This session has no sequence numbers to send or to compare. The primary half session indicates this by resetting the sequence number indicator bit in the BIND request. If the BIND is negotiable, the secondary half session indicates this by the same bit setting in the BIND response. If STSN is required (as for nonnegotiable BIND), the primary half session sends a SET AND TEST action code with zeros as the sequence number values.
- NOT PENDING

  This session has no work units that are indoubt (that is, waiting to be committed)
- PENDING

  One work unit is in doubt (waiting to be committed). The sequence numbers available to the session are the committed number (the sequence number of the last message to be committed) and the potential number (the sequence number of the last message to be issued).
- DECISION TO COMMIT

  An indoubt work unit is committed during the session outage. Potential and committed numbers are still available. (A decision to commit occurs only as the result of a /DEQ command being issued during a session outage.)
- DECISION TO BACK OUT

  An indoubt work unit is backed out during the session outage. Potential and committed numbers are still available. (IMS does not back out work units, but other session partners might).
- INVALID

  Invalid sequence numbers are detected. (An invalid sequence number is one that should not occur, as, for example, a sequence number sent on the secondary-to-primary flow that does not match the committed or potential numbers saved by the STSN receiver.) This type of condition usually indicates an incorrect log data set; the master terminal operator must intervene to recover the session.
- RECEIVED PENDING (Sync-Point Response Lost)

  The receiver of this flow receives the entire work unit that the sender sends and generates a DR2 that might be lost in the network.
- NOT RECEIVED PENDING (Sync-Point Response Not Lost)

  The receiver of this flow never receives the RQ*2 that the sender sends and therefore backs out the work unit that the sender is pending on.

The half sessions exchange information on their respective states by using bind negotiation and the action codes in the STSN request and response.

Related reference

"Using STSN to resynchronize sessions" on page 500

The tables in the following topics show the results and actions related to using STSN for primary-to-secondary (P-S) and secondary-to-primary (S-P) flows.

Completing session initiation

After the half sessions are in sync, either half session can decide to accept or reject the session.

- If the primary half session does not want to continue the session, it sends UNBIND.
- If the primary half session wants to continue the session, it sends start data traffic (SDT).
  - If the secondary half session responds positively, the session is formally bound and traffic on the session can begin.
  - If the secondary half session wants to discontinue the session, it rejects the primary half session's SDT, whereupon the primary must respond with an UNBIND and the session terminates.

Session initiation, including allocation of the LTERM subpool, is completed when a start data traffic (SDT) response has been received if IMS is the primary half session or when an SDT response is sent if IMS is the secondary half session. The LTERM subpool remains allocated to the other half-session name, even across session or subsystem failures, until the session is terminated by mutual consent of both half sessions (symmetrical shutdown). This requires that, after they have been allocated, all subsequent session binds for the same half session must specify the same bind session qualifiers that were active at abnormal session termination until the subpool is released through normal session termination.

If the session is terminated prior to the completion of SDT, any newly attempted subpool allocation caused by a session cold start is backed out, and the subpool is returned to an available-for-allocation status.

In IMS, the master terminal operator must always be notified of any session that is rejected prior to a start data traffic (SDT) completion. In an ISC session, notification to the IMS master terminal operator is optional for normal initiation and termination sequences. Operator notification is specified by the OPTIONS keyword parameters MTOMSG and NOMTOMSG on the TYPE and TERMINAL macros or by an ETO logon descriptor.

Running the session

Use the data flow control protocols to control the flow of data in an ISC session.

Related concepts

“Handling IMS response mode or conversational output errors” on page 505
This topic describes how IMS handles response and conversational mode errors during an ISC session and how to keep the half sessions in sync.

Related reference

“Data flow control protocol reference” on page 515
The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

Terminating an ISC VTAM session

Terminating an ISC VTAM session releases a logical unit from its current connection to the VTAM application program, making the LU available for sessions with other VTAM applications, or terminating communications altogether.

Definitions: The two types of session termination are *normal* and *abnormal*. Normal termination allows both half sessions to complete normal processing before the session is terminated. Abnormal termination forces the session to terminate unconditionally.

Session termination can be invoked by the IMS master terminal operator, the VTAM network operator, or either half session.

Because of this variety of session termination methods, each IMS network installation must determine specific procedures for session termination. When developing these procedures, consider the requirements for session-termination processing.

Normal termination

Normal termination of an ISC session occurs with the flow of the data flow control indicators stop bracket initiation (SBI) and bracket initiation stopped (BIS). Normal termination can be initiated by either half session.

Normal session termination also occurs when the MTO invokes an orderly termination of the IMS network by the IMS **/CHECKPOINT** command with the **FREEZE**, **PURGE**, or **DUMPQ** parameter and the **QUIESCE** parameter. The **QUIESCE** parameter ensures that message queues are emptied before the session is terminated.

When all terminals have indicated that shutdown is complete, IMS:

- Performs checkpoint processing and then issues the VTAM CLSDST macro instruction, when acting as primary half session.
- Awaits **UNBIND**, if acting as secondary half session.

CLSDST causes VTAM to send the **UNBIND** command. **CLSDST** or **UNBIND** releases the logical units from session with IMS. Any further data transmission to the logical units is prohibited.

During the processing of an orderly session termination, the IMS master terminal operator can terminate the network unconditionally by using an IMS **/CLSDST**, **/STOP**, or **/CHECKPOINT** command.

Abnormal termination

Abnormal session termination can occur as a result of transmission or protocol errors, or errors in data that make that data unacceptable to the receiving message processing program. Because an ISC session involves two peer-level systems, error recovery processing and abnormal session termination processes can differ. IMS-detected error conditions requiring abnormal session termination result in IMS issuing a VTAM CLSDST macro when IMS is the primary half session, or TERMSESS with OPTCD=UNCOND when IMS is the secondary half session.

Related reference

“Symmetrical session shutdown for LU 6.1 (SBI and BIS)” on page 540
Two data flow control commands allow a symmetrical and orderly termination for peer level LU 6.1 half sessions: stop bracket initiation (SBI) and bracket initiation stopped (BIS).

“Data flow control protocol reference” on page 515
The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

Using STSN to resynchronize sessions

The tables in the following topics show the results and actions related to using STSN for primary-to-secondary (P-S) and secondary-to-primary (S-P) flows.

Primary-to-secondary flow matrix

The following table illustrates the actions taken when resynchronizing a session and the STSN command flows from the primary half session to the secondary half session.

The first result in each table cell represents the STSN command action; the second result represents the response returned to the STSN command. These are separated by a slash (/). When two STSNs are sent, they are separated by a comma.

Table 83. STSN primary-to-secondary flow

Primary half session	Secondary half session, session cold	Secondary half session, not received: pending	Secondary half session, received: pending
Pending	1 Set and Test / Reset	2 Set and Test / Negative	3 Set and Test / Positive
Decision to commit	4 Set and Test / Reset	5 Set and Test / Negative, Set / Positive	6 Set and Test / Positive
Decision to back out	7 Set and Test / Reset	8 Set and Test / Positive	9 Set and Test / Negative, Set/Positive
Not pending	10 Set and Test / Reset	11 Set and Test / Positive	Not Applicable

Notes to the preceding table

1 The STSN sender is PENDING. The STSN receiver has no sequence numbers. A COLD/WARM mismatch has occurred. The session can continue. The STSN sender should continue the session.

2 The STSN sender is PENDING. The STSN receiver never received the pending RQ*2 and returns a TEST NEGATIVE in the STSN response. The STSN sender backs out the pending work unit to the last commit point.

3 The STSN sender is PENDING. The STSN receiver received the pending RQ*2 and sent a + DR2. The STSN receiver responds TEST POSITIVE, and the STSN sender commits the pending work unit.

4 The STSN sender wants to release the locks and commit resources; however, an RQ*2 is outstanding. The STSN receiver has no sequence numbers. A COLD/WARM mismatch has occurred. The session can continue. The STSN sender should continue the session.

5 The STSN sender wants to release the locks and commit resources; however, an RQ*2 is outstanding. The STSN receiver never received the pending chain, as indicated by TEST NEGATIVE in the STSN response. The STSN sender sends a second STSN with the SET action code to inform the STSN receiver that a wrong decision was made. The STSN receiver can continue the session by sending TEST POSITIVE or decline by sending INVALID.

6 The STSN sender wants to release locks and commit resources; however, an RQ*2 is outstanding. The STSN receiver received the pending chain and committed it, as indicated by the TEST POSITIVE in the STSN response.

7 The STSN sender wants to back out the pending work unit, but a pending RQ*2 is outstanding. The STSN receiver has no sequence numbers. A COLD/WARM mismatch has occurred. The session can continue. The STSN sender should continue the session.

8 The STSN sender wants to back out the pending work unit, but an RQ*2 is outstanding. The STSN receiver never received the pending chain, so the sequence numbers match. A TEST POSITIVE is returned on the STSN response.

9 The STSN sender wants to back out the pending work unit, but an RQ*2 is outstanding. The STSN receiver received the pending chain and committed resources. Because the STSN sender sent out the old committed sequence numbers and the STSN receiver received the chain, a TEST NEGATIVE is returned. The STSN sender informs the STSN receiver that a wrong decision has been made by sending a second STSN with a SET action code. The STSN receiver can continue the session by sending TEST POSITIVE or decline by sending INVALID.

10 The STSN sender is not pending on the P-S flow. The STSN receiver has no sequence numbers. The session continues.

11 The STSN sender is not pending on the P-S flow. The STSN receiver agrees with the sequence numbers on the STSN and returns a TEST POSITIVE.

Secondary-to-primary flow matrix

The following table illustrates the actions taken when resynchronizing a session and the STSN command flows from the secondary half session to the primary half session.

The first result in each table cell represents the STSN command action; the second result represents the response returned to the STSN command. These are separated by a slash (/). When two STSNs are sent, they are separated by a comma.

Table 84. STSN secondary-to-primary flow

Primary	Secondary, session cold	Secondary, pending	Secondary, decision to commit	Secondary, decision to back out	Secondary, invalid	Secondary, not pending
Not received: pending	1 Set and Test / Reset	2 Set and Test/ Positive	3 Set and Test/ Negative	4 Set and Test/ Positive	5 Set and Test / Invalid	6 Set and Test / Positive
Received: pending	7 Set and Test / Reset	8 Set and Test/ Positive	9 Set and Test/ Positive	10 Set and Test/ Negative	11 Set and Test / Invalid	Not Applicable

Notes to the preceding table

1
The STSN sender is not COLD. The STSN receiver has no sequence numbers. The session continues.

2
The STSN receiver has a pending work unit and has held onto locks and not committed resources. The sequence number sent on the SET AND TEST action code indicates that the STSN sender did not receive the pending RQ*2. Therefore, the STSN receiver backs out the pending work unit to the last commit point and returns a TEST POSITIVE on the STSN response.

3
The STSN receiver had a work unit pending and decides to commit resources and release locks. The STSN sender did not receive the RQ*2 sent by the STSN receiver. The STSN receiver returns a TEST NEGATIVE on the STSN response to inform the STSN sender that a wrong decision was made. The STSN sender can continue the session by sending SDT or decline by sending UNBIND.

4
The STSN receiver had a work unit pending and decides to back out the pending work unit. The STSN sender never received the pending work unit, and the STSN receiver returns a TEST POSITIVE on the STSN response.

5
The STSN receiver detects a severe loss of synchronization (possible log data set mismatch) and returns an INVALID on the STSN response.

6
The STSN receiver has no pending work unit and agrees with the sequence numbers sent on the STSN; therefore, it replies with a TEST POSITIVE.

7
The STSN sender is not COLD. The STSN receiver has no sequence numbers. The session continues.

8
The STSN receiver has a pending work unit and still holds resource locks. The number sent by the STSN sender on the S-P SET AND TEST indicate to the STSN receiver that the STSN sender

received the pending work unit. The STSN receiver commits the pending work unit and responds TEST POSITIVE.

9
The STSN receiver had a pending work unit and decides to commit the pending work unit. The number sent by the STSN sender on the S-P flow indicates that the STSN sender received the RQ*2 and a TEST POSITIVE is sent on the STSN response.

10
The STSN receiver had a pending work unit. The receiver decides to back out that work unit and release the locks. The number sent by the STSN sender on the S-P flow indicates to the STSN receiver that the RQ*2 was received and processed by the STSN sender. The STSN receiver indicates the wrong decision by a TEST NEGATIVE on the STSN response. The STSN sender can continue the session by sending SDT or decline by sending UNBIND.

11
The STSN receiver detects a severe loss of synchronization (possible log data set mismatch) and indicates this to the STSN sender by an INVALID in the STSN response.

STSN command format

Both the STSN command and STSN response contain a 5-byte data field.

The format of the STSN command is:

Byte 0
Action code

Bytes 1, 2
Sequence number of the last inbound sync point message sent to the PHS

Bytes 3, 4
Sequence number of the last outbound sync point message sent from the PHS

The primary half session uses the action code to ask the secondary half session to verify the VTAM sequence numbers. The bits of the action code byte are as follows:

Bits 0 and 1
Refer to the inbound sequence-number field

Bits 2 and 3
Refer to the outbound sequence-number field

Bits 4 through 7
Reserved

These values are acceptable for bits 0, 1, 2, and 3 of the STSN command action code:

00 IGNORE
Do not alter value. IMS does not send this action code value as PHS. IMS as SHS returns an INVALID response code for this action code when resynchronization is required and TEST POSITIVE when resynchronization is not required.

01 SET
Set the appropriate sequence number to the value indicated in the sequence number field. For ISC, this code only occurs when the PHS must send a second STSN to indicate a unilateral decision was made to commit or back out a pending work unit.

10 SENSE
Do not alter value. The SHS should return its version of the sequence number in the command response. IMS as SHS returns an INVALID response to this action code.

11 SET AND TEST
Set the appropriate sequence number to the value indicated in the sequence number field. For ISC, this code always occurs for both flows on the first STSN sent by the PHS. The SHS must indicate in the command response whether the sequence number values are acceptable.

When the secondary half session receives the STSN command, it must be able to:

- Verify the SHS-outbound (PHS-inbound) sequence number and arrange to retransmit a message to the PHS if required.
- Verify the SHS-inbound (PHS-outbound) sequence number and inform the PHS whether the number matches the SHS-saved number. Further, the SHS might or might not agree with unilateral decisions by the PHS to commit or back out a pending work unit from the previous session.
- Return to the PHS a DR1 and a 5-byte data response to the STSN command.

The format of the STSN response has the same format as the STSN command:

Byte 0
Action code

Bytes 1, 2
Sequence number of the last recoverable message the SHS sent to the PHS

Bytes 3, 4
Sequence number of the last recoverable message the SHS received from the PHS

The secondary half session (SHS) uses the action code to indicate the test results. The action code must be returned to the primary half session (PHS). Returning the sequence numbers (bytes 1 through 4) is optional. However, for debugging recovery and restart problems, always return these to the PHS whenever the STSN numbers do not match those maintained by the SHS. The bits of the action code byte are as follows:

Bits 0 and 1
Refer to the SHS inbound sequence number

Bits 2 and 3
Refer to the SHS outbound sequence number

Bits 4 - 7
Reserved

These values are acceptable for bits 0, 1, 2, and 3 of the STSN response action code:

00 RESET
Returned on both sequence number flows in response to the SET AND TEST option to indicate that the SHS must cold start and has no sequence number information. The PHS should continue session initiation and should not treat this response as an error. IMS sends this code when cold-starting and replying to STSN, and continues the session when receiving it.

01 TEST POSITIVE
Returned in response to the SET AND TEST option to indicate that the sequence number agrees with the sequence number that the SHS checkpointed. This code should be returned in response to the SET option when the SHS agrees to continue session initiation following a unilateral decision by the PHS to commit or back out a pending work unit on its outbound flow during the session outage.

10 INVALID
Returned in response to the SET AND TEST option to indicate that the STSN SHS outbound sequence number does not agree with the sequence number that the SHS checkpointed (major session restart mismatch). INVALID is also returned in response to the SET option to indicate disagreement with a unilateral action by the PHS to commit or back out a pending work unit on its outbound flow. The PHS should not continue the session when receiving an INVALID response to either flow.

11 TEST NEGATIVE
Returned in response to the SET AND TEST option on the SHS inbound flow to indicate that the sequence number does not agree with the sequence number checkpointed by the SHS. The PHS responds with SDT if the reason for the mismatch is a pending work unit sent by the PHS, but is not received by the SHS; with a second STSN indicating the SET option for both flows if the PHS had made a unilateral decision to commit or back out a pending work unit; or UNBIND if no work unit was pending and no unilateral decision had been made (major session restart mismatch). TEST NEGATIVE can also be returned in response to the SET AND TEST option to indicate that the SHS had made a unilateral decision to commit or back out a pending work unit on its outbound flow. In this case,

the PHS can optionally agree to continue session initiation by sending SET or not to continue session initiation by sending UNBIND.

Related tasks

"Controlling unilateral decisions about pending work units" on page 493
To prevent resources from accidentally getting out of synchronization between subsystems, you can specify whether to continue session resynchronization after a session outage, where the other half session has made a unilateral decision to commit or back out a pending unit of work.

Handling IMS response mode or conversational output errors

This topic describes how IMS handles response and conversational mode errors during an ISC session and how to keep the half sessions in sync.

IMS messages cannot be canceled by session termination or protocols after they have been received and enqueued (made available for scheduling). However, errors can be detected by either half session after the input message has been enqueued. If the error is detected by IMS while processing the transaction, but before the reply message is committed (for example, a DFS555 transaction abend occurs), both half sessions are able to back out, because IMS holds the input sync-point response until the reply message is available for output. An exception response and appropriate error recovery process (ERP) message are returned. Errors occurring after the reply message is made available for output result in backout to the last application sync point (the one that made the reply message available). These errors are not communicated to the other half session.

After an exception response to a DFC command, you might need to purge before further sends or receives.

Related reading: For information on the DFC protocols, see *z/OS Communications Server: SNA Programming*.

Related reference

"Data flow control protocol reference" on page 515
The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

"ERP purging" on page 525
After sending an exception response and before continuing to send or receive, the exception response sender might need to enter error recovery process (ERP) PURGE mode until the DFC state managers of both half sessions are synchronized.

Response mode errors

Errors detected by the other half session cannot be communicated to IMS until IMS attempts to send the reply.

At this point, errors must be communicated to IMS before the requested sync-point response is returned, because all response mode output is sent indicating RQD and EB. Errors are reflected by the return of an exception response with an optional ERP message, which might contain appropriate sense data and protocols.

The exception response or ERP FM header sense codes can cause the output message to be backed out (dequeued) or retransmitted, or they can cause the session to terminate with the message still on the queue. ERP message protocols can cause the message to be backed out (dequeued) regardless of the sense code through use of EB or can leave IMS in send state through use of CD. This latter case is only allowed if the sense code used results in the output message being dequeued or retransmitted.

An LUSTATUS - abort with EB is sent as the next output on the session if CD is used on the ERP message and the sense code results in dequeuing the response mode output message. However, in all of the cases where the error is detected by the other half session, the reply message is the only resource that IMS can back out. The database updates and other messages initiated by the response mode transaction cannot be backed out by IMS.

Conversational mode errors

Errors detected by the other half session cannot be communicated to IMS until IMS attempts to send the output reply.

Errors must be communicated to IMS before the requested sync-point response is completed. For nonlast conversational output, the sync-point requested using RQE2 with CD and any returned normal flow data implicitly completes the sync point. The last conversational output message is sent requesting RQD2 with EB. Errors are reflected either by returning an exception response with an optional ERP message (which might contain appropriate sense data and protocols) or by returning an LUSTATUS with appropriate sense codes and protocols.

The sense codes can cause the message to be backed out (dequeued) or retransmitted, or they can cause the session to terminate with the message still on the queue. ERP and LUSTATUS message protocols can cause the message to be backed out (dequeued) using EB, or they can leave IMS in send state using CD. This latter case is only allowed if the sense code retransmits the message or if the last conversational output for the sense code dequeues the output message. An LUSTATUS - `abort` with EB is sent as the next output on the session if CD is used on the LUSTATUS or ERP message, and the sense code results in dequeuing the last conversational mode output message.

When the error is detected by the other half session, the output reply message is the only resource that IMS can automatically back out. The database updates, conversational SPA, and other messages initiated by the conversational transaction cannot be automatically backed out by IMS. However, for conversations, an internal IMS **/EXIT** command is scheduled to invoke the user's Conversational Abnormal Termination exit routine, which can optionally schedule a transaction to reverse necessary database changes based on the conversational SPA content. The interface to the exit routine is the same as if an **/EXIT** command had been received on the session.

Related reference

Conversational Abnormal Termination exit routine (DFSCONE0) (Exit Routines)

Normal conversation termination extension with ISC

In a non-ISC environment, normal termination of IMS conversation mode occurs when the IMS transaction creates a blank in the transaction-code field of the conversational SPA prior to committing the reply message. The conversation ends when the reply message has been successfully dequeued.

An extension for normal termination has been made for ISC, because the peer-level application in the other half session can also now end the conversation. However, IMS supports this request for normal termination by receiving only an SNA-defined commit request - LUSTATUS X'0006' with EB. This occurs because of a stand-alone commit request (or normal termination) by the remote application to its subsystem. This form of conversational termination requires notification of the user using the Conversational Abnormal Termination exit routine (new input vector of X'28' in byte 3 of register 1), because committing changes might have been deferred by the completed conversational steps. This is possible by recording the deferred information within the conversational SPA. The exit routine can schedule an appropriate transaction to commit the deferred changes upon being invoked with the new vector.

Restriction: IMS does not support an input message with EB to terminate the conversation.

Keeping half sessions synchronized

Sync-point responses (DR2) are used between ISC session partners to ensure that both partners' sync-point managers can commit or back out recoverable resources synchronously.

All messages sent or received on an IMS ISC session are defined as either recoverable or irrecoverable, depending on the message type. The session-response protocols are used to ensure that both ends of an ISC session mutually understand the recoverability attributes associated with each message. The response protocol used must be consistent with the IMS message type.

The ISC sync point can be explicit or implicit:

Explicit
The input requests an RQD2 response.

Implicit
The input requests the sending of an exception response (RQE2) and change-direction on output made available by the transaction, or the sending of an LUSTATUS in lieu of output data from the transaction).

To further increase integrity and recoverability, IMS can supplement the sync point facility by logging the information. Use of log write-ahead ensures that the sync point indication is recorded on the log. This makes sync point information available to IMS restart procedures before the sync-point response is actually sent or change-direction is replied to (implied sync point).

Before reading these topics, be sure to understand the definitions and relationships of the ISC sync point and associated commit and backout processes to the IMS application program, described in these paragraphs.

Related information
VTAM and SNA reference information (System Programming APIs)

Sync points requested on input to IMS

For any type of input, IMS does not schedule the intended transaction until the complete input message is successfully received.

Sender-detected errors, errors resulting from processing of IMS input, and session failure prior to the receipt of the complete input message cause the entire message to be discarded or backed out. However, the input message cannot be canceled by ISC session failures or protocols after the complete message is received and made available for scheduling.

When an input message is backed out because of errors detected during IMS input processing or during synchronous transaction execution, the session partner is notified, either by means of session termination or by an exception response to the ISC input. These events occur for this backout:

- Backout results in resetting the associated DFC and ATTACH states to those of the last sync point.
- Backout during transaction execution results in backing out application updates and messages, except express messages made after the last application sync point.
- The application sync point and ISC sync point are not necessarily the same.
- Backout during input to IMS has no effect on other recoverable IMS resources, such as databases, because input messages are not available for scheduling or execution until they are received completely and without error.
- The result of the backout is only the current input message, even if several consecutive input (irrecoverable) messages were received and executed or enqueued for scheduling after the last input sync point was requested from IMS.

The definition and relationship of successful ISC input sync points to IMS application sync points depend upon whether the ISC input was synchronous or asynchronous. For asynchronous input, the sync point reflects only that IMS is now responsible for message recovery. No implications exist relative to the scheduling or execution of transactions or to the availability of transaction output. After the message is successfully enqueued, the DFC and ATTACH sync point information is updated and the requested sync-point response is returned to the session partner.

Although some IMS exceptions for synchronous input exist, the sync point is intended to reflect the success of the IMS transaction execution and sync point. IMS updates the DFC and ATTACH sync point information as appropriate and commits all associated transaction resources (for example, DL/I databases) and the output transaction message reply when responding to the ISC attached input sync-point request.

These exceptions apply to sync points for synchronous input to IMS:

- The ISC sync-point response is returned by IMS when a transaction-inserted response mode or conversational reply message (first message inserted to the I/O PCB or alternate response PCB) is made

available for output using a transaction sync point. Additional transaction processing and sync points are not reflected in the ISC session. No ISC sync point-response occurs for transaction sync points prior to the transaction's inserting the reply message, as in a program-to-program switch. Nonfirst messages inserted to the I/O PCB or alternate response PCB, messages inserted to nonresponse PCBs, and IMS express messages intended for the ISC session are queued for future asynchronous delivery after successful delivery of the reply message.

- Except during program-to-program switches to another conversational transaction, IMS generates a subsystem error message as the result of abnormally terminating a conversational transaction that attempts to cause a transaction sync point without first having inserted an output conversational reply message. The subsystem error message causes an exception response to be sent to the input sync-point request and the input message to be backed out.

Sync points requested on output by IMS

IMS commits the output message when the requested sync-point response is returned by the other session partner. The message might also be committed as the result of some sense codes that are returned on an exception response to the requested sync-point response.

Definition: The term *commit* means that the message has been successfully sent and dequeued, and sync point information has been updated as appropriate.

Use of the IMS /DEQUEUE command during a session outage while an output sync-point response is pending is considered a unilateral decision by an authorized terminal operator to commit the sending output message.

Depending upon the sense code used, IMS backs out the output message when an exception response is returned to the sync-point request or when the IMS /DEQUEUE command is issued by an authorized terminal operator before IMS requests a sync-point response.

Definition: The term *backout* means that a recoverable message is returned to the message queue (unless dequeued by a /DEQUEUE command) for subsequent retransmission. An irrecoverable message is either dequeued or returned to the message queue for subsequent retransmission, depending upon the type of error. Normally, an exception response or IMS failure causes an irrecoverable message to be dequeued. Some session failures that do not result from a subsystem failure cause an irrecoverable message to be retransmitted at the first opportunity after resynchronization. Backout results in resetting the associated DFC and ATTACH states to those of the last sync point.

Sync point and response requirements

The IMS input/output message flow can be represented as an input/output flow from a sequential queue data set.

In order to maintain integrity and recoverability of this queue data set, IMS requires that each recoverable input and output message establish a sync point between both half sessions before continuing the flow. This allows both half session sync-point managers to commit or back out resources in synchronism.

The sync point facility for ISC redefines and separates the DR1 and DR2 requests and responses. The DR2 requests and responses are known as sync-point requests and responses and are functionally independent from those associated with DR1.

Recoverable messages

To ensure that a recoverable transaction can be recovered, IMS requires the following response protocols for each recoverable message sent or received.

- Messages that are not MFS-paged messages:

  An exception DR2 (RQE2) must be requested on each nonlast (or when it is not the only) RU of an SNA chain.

  Except when change-direction is sent, each last or only RU of an SNA chain must request a DR2 (RQD2). Either exception DR2 (RQE2) or RQD2 can be requested when change-direction is sent.

A sync-point response on the last or only RU of an SNA chain always indicates end-of-message. End-of-message always occurs at end-of-chain for single chain (non-MFS-paged) messages.

- First chain of asynchronous (ATTACH SCHEDULER) demand-paged message:

  The first chain (OIC) of asynchronous demand-paged messages is an ATTACH for the SCHEDULER model. This OIC requests a DR2 (RQD2) with end-bracket. This allows the application to be asynchronously scheduled to receive the message and leaves the session in a state to be allocated to the scheduled application.

- Nonlast pages of MFS demand-paged (output using ATTACH or ATTACH SCHEDULER) messages, and last pages of MFS-operator logical-paged (OLP) output:

  An exception DR1 (RQE1) is requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS demand-page).

  Each last or only RU of an SNA chain (MFS demand-page) requests exception DR1 (RQE1) with change-direction.

- Last pages of MFS demand-paged (output) messages:

  An exception DR2 (RQE2) is requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS demand-page).

  Except when change-direction is sent on the last page, each last or only RU of an SNA chain (MFS demand-page) requests a DR2 (RQD2). Exception DR2 (RQE2) with change-direction is requested.

  A sync-point response on the last or only RU of an SNA chain always indicates end-of-message. A sync point must always be requested on the last page of MFS demand-paged messages to ensure end-of-message.

- First, nonlast pages of MFS-autopaged messages:

  An exception DR1 (RQE1) is requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS autopage) sent to IMS and must be requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS autopage) received by IMS.

  RQD1 is requested on the last or only RU of the first SNA chain of an MFS-autopaged output message sent by IMS.

  RQD1 must be requested on the last or only RU of the first SNA chain of an MFS-autopaged input message received by IMS when the other half session is initialized as the primary half session (bidder). Either exception DR1 (RQE1) or DR1 (RQD1) can be requested on the last or only RU of this SNA chain when the other half session is initialized as the secondary half session (first speaker). Change-direction is not sent and must not be requested on nonlast pages of MFS-autopaged input and output messages.

- Nonfirst, nonlast pages of MFS-autopaged messages:

  An exception DR1 (RQE1) is requested on each RU (including last or only) of an SNA chain (MFS autopage) sent by IMS. An exception DR1 (RQE1) must be requested on each nonlast (or when it is not the only) RU of an SNA chain received by IMS. Either exception DR1 (RQE1) or RQD1 can be requested on each last or only RU of an SNA chain received by IMS. Change-direction is not sent and must not be requested on nonlast pages of MFS-autopaged input and output.

- Last pages of MFS autopaged messages:

  An exception DR2 (RQE2) is requested on each nonlast RU of an SNA chain (MFS autopage) sent by IMS and must be requested on each nonlast RU of an SNA chain (MFS autopage) received by IMS.

  Except when change-direction is sent on the last page, each last or only RU of an SNA chain (MFS autopage) requests a DR2 (RQD2). Exception DR2 (RQE2) is requested when change-direction is sent.

  Except when change-direction is received on the last page, each last or only RU of an SNA chain of an MFS-autopaged input must request a DR2 (RQD2). Either exception DR2 (RQE2) or definite response 2 (RQD2) can be requested when change-direction is indicated.

  A sync-point response on the last or only RU of an SNA chain or on an LUSTATUS `- commit` always indicates end-of-message. A sync point must always be requested on the last page or on an LUSTATUS `- commit` following the last page of autopaged messages to ensure end-of-message.

For the case when allowing RQE1 on the last or only RU of an SNA chain that does not indicate a change-direction for MFS-autopaged input or output, an exception to the preceding protocols occurs when the session bind indicates DEFINITE RESPONSE CHAINS. If the definite response chains parameter is set for the IMS half session, each last or only RU defined that does not also indicate change-direction is sent requesting RQD1 or RQD2. RQD1 or RQD2 must be requested by the other half session under these same conditions if its session bind indicates DEFINITE RESPONSE CHAINS. Both exception and definite response (DR1 and DR2) are valid if change-direction is indicated under definite response chain rules.

Related reference

“LUSTATUS protocol” on page 528
IMS sends and receives LUSTATUS as summarized in the following table. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this topic.

Irrecoverable messages

IMS treats an irrecoverable message in the same manner as a recoverable one, except that all processing required to achieve recoverability is eliminated. As a result, irrecoverable messages require less processing time, but can be lost in the event of a failure.

Irrecoverable, non-MFS input and output messages basically have the same requirements as those for recoverable messages, except that DR1 and exception DR1 can optionally be requested instead of DR2 and exception DR2, respectively. Irrecoverable MFS-paged input and output messages have the same requirements as recoverable ones.

When messages are being sent, only one message can be outstanding at a time. This means that the sender can send one message and must wait for the response or reply before sending another.

The required response and sync-point protocol allows message integrity to be maintained by allowing the half sessions' sync-point managers to mutually understand when messages are accepted (committed) or rejected (backed out). This also allows change-direction to be solicited using SIGNAL RCD as required. RQE1 or RQE2 is recommended where change-direction is indicated, because these capabilities are automatic. An RQD1 or RQD2 are valid, but reduce performance because of the unnecessary response. A response or sync point is implied when a reply is received to a sent message indicating change-direction and RQE1 or RQE2. That is, the reply is an implied DR1 or DR2 response. Requesting either a definite response (RQD1 or RQD2) or an exception response (RQE1 or RQE2) with change-direction is valid for the session bind option DEFINITE RESPONSE CHAINS.

A failure can occur between sending a recoverable message and receiving the sync-point response (or reply) and receiving the sync-point response (or reply message). During the session restart procedure, the STSN command is used to inform both half sessions of the sequence number of the last sent or received sync-point message. If either half session has not completely received a given message, that message can then be retransmitted.

Sync-point indicators on messages

The following topics describe the sync-point indicators sent on IMS input and output messages.

Requests on IMS input messages

The response and sync-point requests for input messages to IMS are summarized in the following table.

In the table, an "X" in the table indicates the entry is supported by IMS. An "S" in the table indicates the corresponding entry is suggested.

Table 85. Response and sync-point requests for IMS input messages

	VTAM indicators with message							
	RQE1 "1" on page 512		RQD1		RQE2 "1" on page 512		RQD2	
Input message type	**CD**	**¬CD**	**CD**	**¬CD**	**CD**	**¬CD**	**CD**	**¬CD**
Nonlast MFS page (autopage)		S		X				
Last MFS page (autopage)								
Recoverable and nonrecoverable					S		X	S
Fast Path conversational recoverable and nonrecoverable transaction								
Response mode transaction	S		X	X	S		X	X
Nonrecoverable, nonresponse, nonconversational mode transaction	S	S"2" on page 512	X	X	S	Note "2" on page 512	X	X
Recoverable, nonresponse, nonconversational mode transaction						S	X	S
MFS paging control request: SNA-formatted QMODEL FMHs	S							
MFS paging control request: SNA RAP FMH					S		X	
IMS message switch								
IMS message switch using ATTACH SCHEDULER					S		X	S
IMS message switch using ATTACH (no SCHEDULER)								Note "3" on page 512
ATTACH SYSMSG"4" on page 512								Note "3" on page 512
IMS command	S		X	X	S		X	X
VTAM command"5" on page 512								
LUSTATUS - commit					S		X	S

Table 85. Response and sync-point requests for IMS input messages (continued)

Input message type	VTAM indicators with message							
	RQE1 "1" on page 512		RQD1		RQE2 "1" on page 512		RQD2	
	CD	¬CD	CD	¬CD	CD	¬CD	CD	¬CD
Other	X		X	X				
Test mode"6" on page 512	S		X	X	S		S	X
FMH7 (ERP) messages	X	Note "3" on page 512	X	Note "3" on page 512				

Notes:

1. A response is implied by a reply to the change-direction indicator; therefore, RQE1 or RQE2 is recommended when change-direction is sent. Either method of requesting a response is supported for the session BIND option of DEFINITE RESPONSE CHAINS. When IMS is running as a secondary system, the other half session (as the primary system), if sending a BB-CD chain, must indicate RQD1 to allow proper DFC bracket and send/receive synchronization.
2. Supported only for irrecoverable-inquiry input requesting EB or BB/EB.
3. Sent with EB. BB is also sent as needed.
4. Because IMS SYSMSG is handled in the same manner as is a message switch, the same protocols apply.
5. LUSTATUS might indicate RQE1 with EB or BB/EB. CHASE, LUSTATUS, and CANCEL commands can optionally request RQE1 with CD. LUSTATUS - commit can also request RQE2 or RQD2 for specific conditions. All other normal flow VTAM commands must request RQD1.
6. Applies to test "echo" mode only. Not applicable to /TEST MFS.

Related reference

"Bracket and half-duplex protocol" on page 515
IMS uses SNA bracket protocol to resolve contention, and uses the change-direction indicator of the half-duplex protocol to control normal flow send/receive mode while within bracket state.

Requests on IMS output messages

The response and sync-point requests for IMS output messages are summarized in the following table.

An "X" in the table indicates that the corresponding entry is supported by IMS.

Table 86. Response and sync-point requests for IMS output messages

Output message types	VTAM indicators with message							
	RQE1"1" on page 514		RQD1		RQE2"1" on page 514		RQD2	
	CD	¬CD	CD	¬CD	CD	¬CD	CD	¬CD
Update, recoverable					X			X
Inquiry, recoverable					X			X
Inquiry, nonrecoverable	X			X				

Table 86. Response and sync-point requests for IMS output messages (continued)

Output message types	VTAM indicators with message							
	RQE1 "1" on page 514		RQD1		RQE2 "1" on page 514		RQD2	
	CD	¬CD	CD	¬CD	CD	¬CD	CD	¬CD
Fast Path (recoverable)								X
Nonlast MFS page (autopaged)		See notes "2" on page 514		See notes "2" on page 514				
First chain demand-paged output								
ATTACH (no SCHEDULER)	X							
ATTACH SCHEDULER								X
Nonlast MFS page (demand-page)	X							
Last MFS page (autopaged or without OLP) recoverable, nonrecoverable					X			X
Last MFS page (with OLP) "3" on page 514	X							
IMS command reply: /DISPLAY, /RDISPLAY, /FORMAT					X			X
IMS command reply: /TEST	X							
Other IMS command replies				X				
Test mode output "4" on page 514	X			X				
Broadcast output (ATTACH SYSMSG)					X			X
System error messages (ATTACH SYSMSG)	X			X	X			X
Message switch output					X			X
VTAM commands "5" on page 514	X			X				

Table 86. Response and sync-point requests for IMS output messages (continued)

Output message types	VTAM indicators with message							
	RQE1 "1" on page 514		RQD1		RQE2 "1" on page 514		RQD2	
	CD	¬CD	CD	¬CD	CD	¬CD	CD	¬CD
FMH7 (ERP) messages	X			See notes "6" on page 514				
QSTATUS (QMODEL) "7" on page 514	X				X			X

Notes:

1. A response is implied by a reply to the change-direction indicator; therefore, IMS indicates RQE1 or RQE2 when sending the change-direction indicator. Either method of requesting a response is supported for the session BIND option DEFINITE RESPONSE CHAINS.
2. RQD1 occurs on each chain rather than RQE1 when IMS is bound as "RQD only," and on the first page of MFS autopaged output to prevent unnecessary ERP overhead if errors or contention is detected by the receiver.
3. MFS operator logical paging is in effect if the MOD specifies PAGE=YES and autopaged output is not indicated in the MFS system control area.
4. Applies to test "echo" mode only. Not applicable to /TEST MFS.
5. CHASE is always sent RQE1/CD.
6. Sent with EB.
7. CD/RQE1 is sent on QSTATUS, which results from an invalid cursor in an MFS DPM demand-page request for output sent as ATTACH SCHEDULER. CD/RQE2 or EB/RQD2 is sent on QSTATUS, which results from the receipt of QPURGE during demand-paged output. CD/RQE2 is sent:
   - When operating in IMS conversation or in test mode
   - When the input is sent with ATTACH SCHEDULER and the associated output component is SINGLE2 or MULT2.

   All other cases result in EB/RQD2.

Related reference

"LUSTATUS protocol" on page 528
IMS sends and receives LUSTATUS as summarized in the following table. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this topic.

"CANCEL protocol" on page 523
IMS sends and receives the SNA **CANCEL** command, which allows the sender to terminate an output SNA chain without having to send all of the chain following a sender- or receiver-detected error.

"Bracket and half-duplex protocol" on page 515

IMS uses SNA bracket protocol to resolve contention, and uses the change-direction indicator of the half-duplex protocol to control normal flow send/receive mode while within bracket state.

Data flow control protocol reference

The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

BID protocol

IMS does not send the SNA BID command, but receives the command when acting as a secondary half session.

When receiving the BID command in a between-bracket state, IMS responds DR1 and enters a receive state for the pending input. If the BID command is received while IMS is in-brackets, IMS rejects the BID with an exception DR1 response (X'08130000').

Bracket and half-duplex protocol

IMS uses SNA bracket protocol to resolve contention, and uses the change-direction indicator of the half-duplex protocol to control normal flow send/receive mode while within bracket state.

IMS uses these options when in session with another ISC logical unit:

- Bracket reset state can be selected at data traffic active state. This state can be set to between-brackets (BETB) or in-brackets (INB).
- Half-duplex send or receive state can be selected at data traffic active state.
- Either half session can send end-bracket (EB). IMS must always be bound to allow EB.
- Half-duplex flip-flop is used for normal flow send/receive states.
- Bracket termination rule 1 (conditional rule) is used.

Related reading:

- For information on bracket termination rules, see *z/OS Communications Server: SNA Programming*.

Related reference

Bind parameters for SLU P and LU 6.1 (System Programming APIs)

Bracket protocol for IMS input

One or more input messages can be received within a bracket.

For example:

- Any number of nonresponse, nonconversation, noncommand, or nontest mode input messages can be sent within the bracket. The input, the subsequent asynchronous output, or an LUSTATUS can end the bracket. Further, a single nonresponse, nonconversation, noncommand, or nontest mode input message with ATTACH EB can be sent.
- One response mode or command message, except /TEST, can optionally occur within the same bracket after any of the input messages listed in the previous list bullet. The response mode or command message output ends the bracket.
- Any number of conversation or test mode inputs followed by conversation or test mode outputs can optionally begin within the same bracket after the input messages listed in the first list bullet. The bracket ends on the last conversation output message (or command complete message from /END for test mode). The bracket can also be ended by the other half session's sending an input LUSTATUS or CHASE with EB, but not by input data sent with EB.
- The bracket can be ended by an EB occurring on an FMH7 or LUSTATUS resulting from an error.

The following series of tables summarizes the bracket and send/receive indicators acceptable for the various message types input to IMS.

An "X" in the table indicates that the corresponding entry is supported by IMS.

In the tables the VTAM indicators are represented by the following abbreviations:

- BB: Begin-Bracket
- EB: End-Bracket
- CD: Change-Direction. Allowed on Last- or Only-in-Chain.
- BB/EB: Begin- and End-Bracket. Allowed on First- or Only-in-Chain.

Table 87. VTAM bracket and send/receive indicators that are sent with messages to IMS: input message type using ATTACH SCHEDULER

Input message type using ATTACH SCHEDULER	BB	EB	BB/EB	BB/CD	CD	¬BB, ¬EB, ¬CD
MFS-autopaged input: First page (of multiple page input)	X					X
MFS-autopaged input: Nonfirst, nonlast page						X
MFS-autopaged input: Last page: see transaction types						
Nonresponse transaction						
Nonconversational mode transaction	Note "1" on page 516	Note "2" on page 516	Notes"1" on page 516, "2" on page 516	Note "1" on page 516	X	X
Response mode (including Fast Path) transaction	Note "1" on page 516			Notes "1" on page 516, "3" on page 516	Note "3" on page 516	X
Conversational transaction	Note "1" on page 516			Notes "1" on page 516, "3" on page 516	Note "3" on page 516	X
IMS message switch	X	X	X	X	X	X
IMS command	X	Note "4" on page 516	Note "4" on page 516	Note "3" on page 516	Note "3" on page 516	X
Input while in test mode"4" on page 516					Note "3" on page 516	X

Notes:

1. Not valid for the last page of MFS-autopaged input.
2. Not valid for the first page of MFS-autopaged input.
3. Denotes the optimal end-of-message indicators to prevent IMS from soliciting change-direction using SNA SIGNAL.
4. Valid only for /DIS, /RDIS, and /FOR commands. Also optimal for these commands to keep IMS from immediately forcing a between-brackets state by sending LUSTATUS - NO-OP (X'0006') with EB. The output from these commands is always queued and sent asynchronously.

5. FMH7 messages and LUSTATUS NO-OP indicating EB force end of conversation, response mode, and test mode and cause the associated IMS output message to be dequeued. Further, the Conversational Abnormal Termination exit routine is invoked.

Table 88. VTAM bracket and send/receive indicators that are sent with input messages to IMS using ATTACH (no SCHEDULER)

Input message type using ATTACH (no SCHEDULER)	BB	EB	BB/EB	BB/CD	CD	BB, EB, CD
MFS-autopaged input: First page (of multiple page input)	X					X
MFS-autopaged input: Nonfirst, nonlast page						X
MFS-autopaged input: Last page: see transaction types						
Nonresponse, nonconversational mode transaction		X	X			
Response mode (including Fast Path) transaction "6" on page 517	Note "1" on page 516			Notes "1" on page 516, "3" on page 516	Note "3" on page 516	X
Conversational transaction	Note "1" on page 516			Notes"1" on page 516, "3" on page 516	Note "3" on page 516	X
IMS message switch		X	X			
ATTACH SYSMSG		X	X			
IMS command	X	Note "4" on page 516	Note "4" on page 516	Note "3" on page 516	Note "3" on page 516	X
Input while in test mode "5" on page 517					Note "3" on page 516	X

Notes:

1. Not valid for the last page of MFS-autopaged input.
2. Not valid for the first page of MFS-autopaged input.
3. Denotes the optimal end-of-message indicators to prevent IMS from soliciting change-direction using SNA SIGNAL.
4. Valid only for /DIS, /RDIS, and /FOR commands. Also optimal for these commands to keep IMS from immediately forcing a between-brackets state by sending LUSTATUS - NO-OP (X'0006') with EB. The output from these commands is always queued and sent asynchronously.
5. Applies to test "echo" mode only. Not applicable to /TEST MFS.
6. For ISC, response mode operation is forced for attached transactions not indicating EB (regardless of their definition on the TRANSACT macro statement) when the TERMINAL macro statement or ETO user descriptor indicates either TRANRESP or FORCRESP. The input is rejected if the TERMINAL macro statement or ETO user descriptor indicates NORESP.

Table 89. Vtam bracket and send/receive indicators sent with other input messages sent to IMS

Other input message types	BB	EB	BB/EB	BB/CD	CD	BB, EB, CD
VTAM normal flow command/ indicator "1" on page 518, "2" on page 518	X				X	X
FMH7 (ERP) messages		Note "5" on page 517	X	X	X	
IMS MFS paging control request (QMODEL FMHs)				X	X	
RAP FMH					X	

Notes:

1. LUSTATUS, CHASE, and CANCEL can indicate EB or CD. Other VTAM normal flow commands or indicators must not indicate either EB or CD.

   When IMS is waiting for conversational input, LUSTATUS and CHASE cannot be sent to IMS unless they also indicate end-bracket. EB causes the conversational output message to be dequeued, the conversation mode to be terminated, and the Conversational Abnormal Termination exit routine to be invoked. The session is terminated if an LUSTATUS or CHASE is received by IMS with any other protocol.
2. FMH7 messages and LUSTATUS NO-OP indicating EB force end of conversation, response mode, and test mode and cause the associated IMS output message to be dequeued. Further, the Conversational Abnormal Termination exit routine is invoked.

Related concepts

"Handling IMS response mode or conversational output errors" on page 505
This topic describes how IMS handles response and conversational mode errors during an ISC session and how to keep the half sessions in sync.

Related reference

"Data flow control protocol reference" on page 515
The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

Bracket protocol for IMS output

The output bracket and send/receive protocol used by IMS and the number of output messages sent per bracket are dependent on a variety of factors.

The output bracket and send/receive protocol used by IMS and the number of output messages sent per bracket depend upon one or more of the following:

- The IMS message type
- Whether the message is to be MFS paged
- Whether the other half session supports the SCHEDULER model
- The output component specified during IMS system definition
- The bracket and ATTACH protocol associated with the originating input transaction

Within IMS, the protocols used for nonlast chains (pages) of MFS-paged output and the last (MFS-paged) or only chain of response mode, conversation mode, test mode, or command replies are predefined regardless of whether the originating input transaction and resulting output replies are synchronous (ATTACH) or asynchronous (SCHEDULER). This is also true for output resulting from input sent with

ATTACH EB and for asynchronous output that must be sent ATTACH because the other half session lacks SCHEDULER support.

CD is used for all nonlast chains of MFS demand-paged messages, the last (MFS-paged) or only chain of test mode, and nonlast conversational mode messages. CD allows the paging operation or synchronous event between IMS and the other half session to continue.

EB ensures that the end of the synchronous event occurs by placing the session in reset state for the last (MFS-paged) or only chain of response mode replies, last conversational mode replies, and command replies. This is also true for replies resulting from input sent with ATTACH EB and for asynchronous output that must be sent ATTACH because the other half session lacks SCHEDULER support. All nonlast chains (pages) of MFS-autopaged messages are sent without either CD or EB.

Using ATTACH SCHEDULER, your installation must define to IMS the protocols to be used for the last (MFS-paged) or only chain of other asynchronous output. IMS allows you to specify single or multiple messages for each component defined for a session and whether IMS should send more than one asynchronous message before indicating change-direction or end-bracket. The possible component definitions are:

SINGLE1
Asynchronous output for this component is sent one message per bracket. Each message begins a bracket (if necessary) and always ends a bracket.

SINGLE2
Asynchronous output for this component is sent one message at a time with the VTAM begin-bracket (if necessary) and change-direction indicators to allow the receiving subsystem to optionally send its communication traffic.

MULT1
All asynchronous messages for a given LTERM are sent before the bracket is ended. Traffic is sent BB (if necessary), message1, message2,...messageN, EB. EB occurs after the last message for the LTERM is acknowledged and dequeued.

MULT2
All asynchronous messages for a given LTERM are sent before change-direction is sent. Traffic is sent BB (if necessary), message1, message2...messageN, CD. CD occurs after the last message for the LTERM is acknowledged and dequeued.

Consider the definition (on the IMS system definition TERMINAL macro statement or an ETO logon descriptor), and use of protocols for ISC components to be used for transmission of asynchronous messages. This is particularly true in an IMS-to-IMS environment, because most messages between the two subsystems are asynchronous. Incorrectly defining or using these ISC protocols can:

- Require extra transmissions to occur in order to acquire the flow or to end the bracket
- Cause unnecessary bracket contention error recovery operations
- Produce output protocols unacceptable to the receiving subsystem.

A further explanation follows:

SINGLE1
Use of SINGLE1 results in EB on each message and might cause bracket contention if the other subsystem has data to send.

EB on each message can cause process errors within the other subsystem if the transaction must reply synchronously within the initiating bracket. An error of this type can occur in the receiving subsystem when a SINGLE1 component is used to send response mode, conversation, or test mode messages, or IMS commands from one IMS subsystem to another.

SINGLE2 or MULT2
Change-direction sent on messages from a component defined as SINGLE2 might result in unnecessary transmissions to end the bracket when either no output is available or no reply is available within the other subsystem.

MULT1 or MULT2
Messages sent on components defined as MULT1 OR MULT2 indicate change-direction or end-bracket only after the last message on a queue has been dequeued. Therefore, extra transmissions might result if the other subsystem must signal for the flow to return synchronous replies.

For components defined as MULT1 and MULT2, IMS suppresses the queue rotation that normally occurs between output messages. This suppression allows all messages from a queue selected for output to be sent before IMS initiates output on other queues. Selection of the next queue with available output occurs when the previous queue has emptied or when input changes the active output queue. The queues are not rotated if an LUSTATUS - queue empty or another input that indicates change-direction occurs for the same queue.

The following tables summarize the bracket and send/receive indicators that are acceptable for IMS output messages.

In the tables the VTAM indicators are represented by the following abbreviations:

INB
Indicates that synchronous or asynchronous output occurs within the same bracket as the previous input

BETB
Indicates that IMS initiates output while in a between-brackets state

BB
Begin-bracket

EB
End-bracket

CD
Change-direction (allowed on last- or only-in-chain)

N
No bracket or send/receive indicators

Table 90. VTAM bracket and send/receive indicators sent with output message using ATTACH SCHEDULER

Message type sent using attach scheduler	**SINGLE1 msg.init INB BETB**	**SINGLE2 msg.init INB BETB**	**MULT1/2****"1" on page 521** **msg.init INB BETB**
MFS demand-paged output:First SNA chain (ATTACH SCHEDULER)"2" on page 521	EBBB/EB	EBBB/EB	EBBB/EB
MFS demand-paged output:Nonlast page			
MFS demand-paged output:Last page if OLP"3" on page 521	CD	CD	CD
MFS demand-paged output:Last page without OLP (See appropriate message type)			
MFS autopaged output: First page	NBB	NBB	NBB
MFS autopaged output: Nonfirst, nonlast page	N	N	N
MFS autopaged output: Last page (See appropriate message type)			
Response mode output (Including Fast Path)	EB	EB	EB
Nonlast conversational output message	CD	CD	CD
Last conversational output message	EB	EB	EB
Message switch output	EBBB/EB	CDBB/CD	NBB

Table 90. VTAM bracket and send/receive indicators sent with output message using ATTACH SCHEDULER (continued)

Message type sent using attach scheduler	SINGLE1 msg.init INB BETB	SINGLE2 msg.init INB BETB	MULT1/2"1" on page 521 msg.init INB BETB
Command output: /FOR, /DIS, /RDIS	EBBB/EB	CDBB/CD	NBB
Command output: /TEST"4" on page 521	CD	CD	CD
Other Command output	EB	EB	EB
Test mode output"4" on page 521	CD	CD	CD
None of the above message types - asynchronous	EBBB/EB	CDBB/CD	NBB

Notes:

1. LUSTATUS - queue empty is used at the end of a queue to send EB for MULT1 and CD for MULT2. EB is sent when an ERP backout has reset the DFC and ATTACH states to between-brackets and no component 1 (COMPT1) is defined during IMS system definition. Also, the BB indicator in the BETB column occurs only for the first message or MFS page that must initiate an output bracket asynchronously.
2. Although IMS is between-brackets after having sent stand-alone ATTACH SCHEDULER for response mode, Fast Path, or conversational demand-paged output, IMS does not accept input until the preceding output is successfully transmitted and dequeued using the appropriate paging requests.
3. MFS operator logical paging (OLP) is in effect if the MOD specifies PAGE=YES and autopaged output is not indicated in the MFS system control area (SCA).
4. Applies to test "echo" mode only. Not applicable to /TEST MFS.

Table 91. VTAM bracket and send/receive indicators sent with output message using ATTACH (no SCHEDULER)

Message type sent using ATTACH (no SCHEDULER)	SINGLE1 msg.init INB BETB	SINGLE2 msg.init INB BETB	MULT1/2"1" on page 521 msg.init INB BETB
MFS demand-paged output: First SNA chain (ATTACH)	CD	CD	CD
MFS demand-paged output: Nonlast page, last page if OLP"3" on page 521	CD	CD	CD
MFS demand-paged output: Last page without OLP (See appropriate message type)			
MFS autopaged output: First page	N	N	N
MFS autopaged output: Nonfirst, nonlast page	N	N	N
MFS autopaged output: Last page (See appropriate message type)			
Response mode output (including Fast Path)"5" on page 522	EB	EB	EB
Nonlast conversational message	CD	CD	CD
Last conversational message	EB	EB	EB
Nonresponse, nonconversational output (No SCHEDULER model defined for other half session)	EB BB/EB	EB BB/EB	EB BB/EB
ATTACH SYSMSG	BB/EB	BB/CD	BB

Table 91. VTAM bracket and send/receive indicators sent with output message using ATTACH (no SCHEDULER) (continued)

Message type sent using ATTACH (no SCHEDULER)	SINGLE1 msg.init INB BETB	SINGLE2 msg.init INB BETB	MULT1/2"1" on page 521 msg.init INB BETB
Command output			
Command output: /TEST"4" on page 521	CD	CD	CD
Other command output (except /FOR, /DIS, and /RDIS)"6" on page 522	EB	EB	EB
Test mode output"4" on page 521	CD	CD	CD

Notes:

1. LUSTATUS - queue empty is used at the end of a queue to send EB for MULT1 and CD for MULT2. EB is sent when an ERP backout has reset the DFC and ATTACH states to between-brackets and no component 1 (COMPT1) is defined during IMS system definition. Also, the BB indicator in the BETB column occurs only for the first message or MFS page that must initiate an output bracket asynchronously.
2. Although IMS is between-brackets after having sent stand-alone ATTACH SCHEDULER for response mode, Fast Path, or conversational demand-paged output, IMS does not accept input until the preceding output is successfully transmitted and dequeued using the appropriate paging requests.
3. MFS operator logical paging (OLP) is in effect if the MOD specifies PAGE=YES and autopaged output is not indicated in the MFS system control area (SCA).
4. Applies to test "echo" mode only. Not applicable to /TEST MFS.
5. Response mode operation always occurs for input transactions that are directly attached.
6. See "LUSTATUS protocol" on page 528 for more information on /DISPLAY, /RDISPLAY, and /FORMAT command output.

Table 92. VTAM bracket and send/receive indicators sent with other output message types

Other output message types	SINGLE1 msg.init INB BETB	SINGLE2 msg.init INB BETB	MULT1/2"1" on page 521 msg.init INB BETB
VTAM command/indicator"2" on page 522			
FMH7 (ERP) messages: During conversation	CD	CD	CD
FMH7 (ERP) messages: During test mode	CD	CD	CD
FMH7 (ERP) messages: Other"3" on page 523	EB BB/EB	CD BB/CD	Note "4" on page 523
QSTATUS (QMODEL)	Note "5" on page 523	Note "5" on page 523	Note "5" on page 523

Notes:

1. LUSTATUS - queue empty is used at the end of a queue to send EB for MULT1 and CD for MULT2. EB is sent when an ERP backout has reset the DFC and ATTACH states to between-brackets and no component 1 (COMPT1) is defined during IMS system definition. Also, the BB indicator in the BETB column occurs only for the first message or MFS page that must initiate an output bracket asynchronously.
2. BIS is sent without CD or EB. CHASE is always sent CD.

3. An FMH7 requests the protocol associated with the resulting between-brackets reset state of the last committed input component after backout of the DFC and ATTACH states due to the ERP operation.
4. FMH7 is sent with EB for MULT1, CD for MULT2.
5. CD/RQE1 is sent on a QSTATUS that results from an invalid cursor in an MFS DPM demand-page request for output. CD/RQE2 or EB/RQD2 is sent on QSTATUS, which results from the receipt of QPURGE during demand-paged output. CD/RQE2 is sent: 1) when in IMS conversation or test mode or 2) when the input is sent with ATTACH SCHEDULER and the associated output component was SINGLE2 or MULT2. All other cases result in EB/RQD2.

Related reference

"ISC data flow control examples" on page 643
The following topics provide examples of ISC data flow control.

"LUSTATUS protocol" on page 528
IMS sends and receives LUSTATUS as summarized in the following table. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this topic.

"CANCEL protocol" on page 523
IMS sends and receives the SNA **CANCEL** command, which allows the sender to terminate an output SNA chain without having to send all of the chain following a sender- or receiver-detected error.

CANCEL protocol

IMS sends and receives the SNA **CANCEL** command, which allows the sender to terminate an output SNA chain without having to send all of the chain following a sender- or receiver-detected error.

The **CANCEL** command that is sent because of either a sender- or receiver-detected error is sent by IMS with the VTAM indicators shown in the following table.

The **CANCEL** command that is received by IMS because of either a sender- or receiver-detected error can indicate whatever protocol (consistent with current DFC states) is appropriate for the message sender.

In the table the VTAM indicators that are sent with the **CANCEL** command are represented by the following abbreviations:

- EB: End-Bracket
- CD: Change-Direction

Table 93. VTAM indicators sent with the CANCEL command

Output (determined in following order)	¬CD or ¬EB	CD	EB
After /DEQ (LUSTATUS ABORT is next)	X		
When primary half session and FIC/OIC was between bracket (conditional BB)	X		
After receiving selective receiver ERP sense code		X	
After receiving SIGNAL		X	
When conversational or response mode output is still available	X		
FIC/OIC was EB RQD* (conditional EB)			X
While in test (echo) mode		X	

Table 93. VTAM indicators sent with the CANCEL command (continued)

Output (determined in following order)	¬CD or ¬EB	CD	EB
Non-MFS demand-paged output for SINGLE1 component	See indicators for FIC/OIC was between bracket or FIC/OIC was EB		
MFS demand-paged output for SINGLE1 component			X
Output for SINGLE2 component		X	
Output for MULT1 or MULT2 component	X		

Chaining protocol

Whether operating as the primary or secondary half session, IMS sends both single and multiple RU chains. Half sessions can operate with either single or multiple RU chains as specified in the bind parameters.

IMS messages are usually sent and received as single SNA chains where only-in-chain (OIC) or first-in-chain (FIC) indicates beginning-of-message, and OIC or last-in-chain (LIC) indicates end-of-message. However, each page of an MFS-paged input or output message is sent or received as a single SNA chain. The beginning-of-message occurs when the first page is sent or received, and the end-of-message occurs at sync point (RQD2 or RQE2, CD) requested at the end of the last input or output page (except for MFS operator logical paging).

Optionally, end-of-message can be requested using LUSTATUS - commit immediately following the last page of MFS DPM-autopaged input.

Use the CANCEL command when errors are detected on multiple RU input chains.

Related reference

“LUSTATUS protocol” on page 528
IMS sends and receives LUSTATUS as summarized in the following table. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this topic.

“ERP purging” on page 525
After sending an exception response and before continuing to send or receive, the exception response sender might need to enter error recovery process (ERP) PURGE mode until the DFC state managers of both half sessions are synchronized.

CHASE protocol

IMS sends and receives SNA CHASE.

IMS sends CHASE requesting RQE1/CD to cause a synchronizing event after the receipt of an exception response when it is between pages (chains) of MFS-autopaged output and is not bound with Definite Response Chains Only.

IMS responds with DR1, as necessary, when receiving CHASE. IMS can receive CHASE between any two messages within a bracket. IMS can also receive CHASE as a synchronizing event after IMS has sent an exception response to MFS-autopaged input. (The sender of the MFS-autopaged input receives the exception response while between output pages or after a chain indicating RQE1.)

Only CHASE carrying EB is accepted while IMS is waiting for conversational input. EB causes the conversational output message to be dequeued, conversation mode to be terminated, and the Conversational Abnormal Termination exit routine to be invoked, just as if an **/EXIT** command had been received. The session is terminated if CHASE is received with any protocol other than EB.

ERP purging

After sending an exception response and before continuing to send or receive, the exception response sender might need to enter error recovery process (ERP) PURGE mode until the DFC state managers of both half sessions are synchronized.

Additionally, ERP PURGE must be complete prior to sending the FMH7 ERP message. ERP purging can occur for either single or multiple SNA chains. Single chain purge occurs when DFC states are synchronized within the same chain receiving the exception response. Multichain purge occurs when more than a single SNA chain must be purged before the DFC state managers are synchronized. (The CANCEL and CHASE commands are logically considered part of the same chain when they occur immediately after the SNA chain that resulted in the exception response.)

Single- and multiple-chain purging occur within the data flow control support layer and are independent of SNA presentation layer functions such as MFS.

ERP PURGE must be entered when a half session sends an exception response to an RU (OIC, MIC, or LIC) that does not result in ending a bracket or in both half-session DFC states being synchronized. That is, ERP PURGE begins when a half session sends an exception response to the following RUs:

FIC RQE* (not applicable for EB)
MIC RQE* (not applicable for CD or EB)
LIC RQE* ¬EB and ¬CD
OIC RQE* ¬EB and ¬CD

ERP PURGE ends when the purging half session receives an RU that either ends the bracket or causes both half-session DFC states to again be synchronized. That is, ERP PURGE terminates when the purging half session receives the following RUs:

OIC RQD*
This includes the CANCEL and CHASE commands, and is not applicable for CD or EB.

OIC RQE* CD or EB
This includes the CANCEL command.

LIC RQD*
Not applicable for CD or EB.

LIC RQE* CD or EB

ERP PURGE can span several SNA chains before terminating. When more than one chain is purged, an additional exception response is necessary at the point of bracket termination or DFC state synchronization (except for CANCEL or CHASE commands) to prevent an ambiguous or incorrect understanding as to the disposition of the message by the sending half session (response receiver). This sender ERP exception response (X'0867') indicates that the chain was purged because of an error on a previous chain and that the message sender should resend the message at the next possible opportunity. The X'0867' exception response must be sent when a half session ends ERP PURGE after receiving the following RUs in a chain subsequent to the one that resulted in the original exception response:

OIC RQD*
This does not include CANCEL or CHASE, and is not applicable for CD or EB.

OIC RQE* CD or EB
Except for CANCEL

LIC RQD*
Not applicable for CD or EB.

LIC RQE* CD or EB

The DFC protocol that can be received with the FMH7 ERP message must be either change-direction or end-bracket and can include begin-bracket, as appropriate. The DFC protocol (CD or EB) that is sent by IMS with the FMH7 ERP message is determined by the ATTDSP value resulting after ERP backout. When backout results in a between-brackets state, the ATTDSP value is component 1. When backout results

in a state other than between-brackets, the ATTDSP value is the last committed input component. The protocols are set for the resulting component. End-bracket is sent when an ERP backout has reset the DFC and ATTACH states to between-brackets and no `component 1` (`COMPT1`) was defined during IMS system definition.

Related reference

“Bracket protocol for IMS output” on page 518
The output bracket and send/receive protocol used by IMS and the number of output messages sent per bracket are dependent on a variety of factors.

Resulting DFC state after sender ERP purge

The following table reflects all valid bracket and send/receive states that result after both half sessions reach a sync point after an exception response that indicates a sender ERP sense code other than selective receiver ERP.

In the table, the following abbreviations are used:

BETB
Between-brackets

INB
In-brackets

PHS
Primary half session

SHS
Secondary half session

Table 94. Resulting DFC states after sender ERP purge

	PHS sends data, SHS sends exception: half-session states		**SHS sends data, PHS sends exception: half-session states**	
Type of chain in error	**PHS**	**SHS**	**PHS**	**SHS**
BB/— FIC				
RQ** CD OIC,LIC	“1” on page 527	“1” on page 527	INB.SEND	INB.RCV
RQD* OIC,LIC	BETB	BETB	INB.RCV	INB.SEND
RQD* CANCEL	BETB	BETB	INB.RCV	INB.SEND
RQ** CD CANCEL	“1” on page 527	“1” on page 527	INB.SEND	INB.RCV
RQD* EB CANCEL	“1” on page 527	“1” on page 527	BETB	BETB
BB/EB FIC				
RQE* OIC,LIC	BETB	BETB	BETB	BETB
RQD* OIC,LIC	BETB	BETB	BETB	BETB
RQD* CANCEL	BETB	BETB	BETB	BETB
RQ** CD CANCEL[1]				
RQD* EB CANCEL[1]				
--/-- FIC				
RQE* OIC,LIC	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.SEND	INB.RCV	INB.RCV	INB.SEND

Table 94. Resulting DFC states after sender ERP purge (continued)

	PHS sends data, SHS sends exception: half-session states		SHS sends data, PHS sends exception: half-session states	
Type of chain in error	**PHS**	**SHS**	**PHS**	**SHS**
RQD* CANCEL	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	BETB	BETB	BETB	BETB
--/EB FIC				
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQD* CANCEL	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	BETB	BETB	BETB	BETB

1. CD and EB might not be sent while in DFC BETB state.

Resulting DFC state after selective receiver ERP purge

The following table reflects all valid bracket and send/receive states that result after both half sessions reach a sync point after an exception response that indicates the SNA selective receiver ERP sense code, and before the FMH7 is sent by the half session detecting the error.

Table 95. Resulting DFC states after selective receiver ERP purge

	PHS sends data, SHS sends exception: half-session states		SHS sends data, PHS sends exception: half-session states	
Type of chain in error	**PHS**	**SHS**	**PHS**	**SHS**
BB/-- FIC				
RQ** CD OIC,LIC	"1" on page 528	"1" on page 528	INB.SEND	INB.RCV
RQD* OIC,LIC	ERP.RCV	ERP.SEND	INB.SEND	INB.RCV
RQD* CANCEL	ERP.RCV	ERP.SEND	INB.SEND	INB.RCV
RQ** CD CANCEL	"1" on page 528	"1" on page 528	INB.SEND	INB.RCV
RQD* EB CANCEL	"1" on page 528	"1" on page 528	ERP.SEND	ERP.RCV
BB/EB FIC				
RQE* OIC,LIC	ERP.RCV	ERP.SEND	ERP.SEND	ERP.SEND
RQD* OIC,LIC	ERP.RCV	ERP.SEND	ERP.SEND	ERP.SEND
RQD* CANCEL	ERP.RCV	ERP.SEND	ERP.SEND	ERP.SEND
RQ** CD CANCEL				
RQD* EB CANCEL				
--/-- FIC				

Table 95. Resulting DFC states after selective receiver ERP purge (continued)

	PHS sends data, SHS sends exception: half-session states		SHS sends data, PHS sends exception: half-session states	
Type of chain in error	**PHS**	**SHS**	**PHS**	**SHS**
RQE* OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	ERP.RCV	ERP.SEND	ERP.SEND	ERP.RCV
--/EB FIC				
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	ERP.RCV	ERP.SEND	ERP.SEND	ERP.RCV

Note:

1. CD and EB cannot be sent while in between-brackets state

LUSTATUS protocol

IMS sends and receives LUSTATUS as summarized in the following table. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this topic.

Any LUSTATUS sense code not listed in the following table causes the ISC session to be terminated. An S in the table indicates the suggested indicator settings for the given LUSTATUS. An X in the table indicates that IMS also supports the specified indicator settings for the given LUSTATUS.

Table 96. VTAM indicators sent with LUSTATUS

	VTAM indicators with LUSTATUS								
	RQE1		RQD1			RQE2	RQD2		
LUSTATUS	**EB**	**CD**	**--**	**CD**	**EB**	**CD**	**--**	**CD**	**EB**
Sense Code Received"1" on page 529 : Commit–X'0006'						S	S	X	S
Sense Code Received"1" on page 529 : NO-OP–X'0006'	X	S		X	S				
Sense Code Received"1" on page 529 : Queue Empty–X'0007'		S		X	S				
Function Abort–X'0864'		S	S	X	S				
Function Abort–X'0865'		S	S	X	S				

Table 96. VTAM indicators sent with LUSTATUS (continued)

	VTAM indicators with LUSTATUS								
	RQE1		RQD1			RQE2	RQD2		
LUSTATUS	**EB**	**CD**	**--**	**CD**	**EB**	**CD**	**--**	**CD**	**EB**
Function Abort–X'0866'		X	X	X	X				
Sense Codes Sent: Commit–X'0006'									
Sense Codes Sent: NO-OP–X'0006'		X			X				
Sense Codes Sent: Queue Empty–X'0007'		X			X				
Function Abort –X'0865'		X	X		X				

Notes:

1. While IMS is waiting for conversational input, it accepts LUSTATUS carrying EB only. EB causes the conversational output message to be dequeued, conversation mode to be terminated, and the Conversational Abnormal Termination exit routine to be scheduled. The session is terminated if an LUSTATUS carrying any other protocol is received.

IMS responds DR1 or DR2 when receiving LUSTATUS RQD1 or RQD2. IMS requests DR1, DR2, or exception DR1 or DR2 as indicated in the previous table. The session is terminated if an exception occurs to LUSTATUS.

- LUSTATUS - Queue Empty.

  IMS sends and receives `LUSTATUS - queue empty`.

  IMS sends `LUSTATUS - queue empty`:

  – After receiving an asynchronous (ATTACH SCHEDULER) input message and having no output immediately available.

  – After sending all available output on a given queue.

  RQD1 and end-bracket are indicated on the LUSTATUS sent when IMS is not in conversation or response mode and has been left in-brackets/SEND, and no output is available to be sent from a component defined as `SINGLE1` or `MULT1`. RQE1 and change-direction are indicated on the LUSTATUS sent when IMS is not in conversation or response mode and has been left in-brackets/SEND, and no output is available to be sent from a component defined as `SINGLE2` or `MULT2`.

- `LUSTATUS - Function Abort`. The supported sense codes are:

  X'0864'
  Loop occurs upon re-execution. Sender should not resend the same data.

  X'0865'
  Data sender is responsible for detecting and preventing loop.

  X'0866'
  Data receiver is responsible for detecting and preventing loop.

  IMS sends `LUSTATUS - function abort`X'0865' only under these conditions:

  – When an IMS **/DEQUEUE** command has been issued by an authorized IMS terminal operator before the last page of an MFS-demand or autopaged output message has been sent by IMS.

    The DFC bracket and send/receive protocol indicated on the LUSTATUS is the same as that which would occur with the last chain (page) had the message been successfully sent.

  The following occurs when IMS receives `LUSTATUS - function abort`:

- Except during conversational mode, any function abort received after an output message (including the last page of MFS demand-paged or autopaged output) sent RQE/CD causes the output message to be committed and does not cause session termination. LUSTATUS that is processed as a normal flow input before it is processed as an LUSTATUS command also causes the message to be committed.
- Any function abort causes an incomplete input MFS-autopaged message to be discarded.
- Function abort X'0864', received before the last page of an output MFS demand-paged message is sent, causes the message to be dequeued prior to continuing normal input/output operations if the output is conversational and demand-paged. The Conversational Abnormal Termination exit routine is invoked, just as if an **/EXIT** command had been received on the session.
- Function abort X'0865', received before the last page of an active output MFS demand-paged message is sent, causes the message to be returned to the output message queue and the session to be terminated.
- Function abort X'0866', received before the last page of an active output MFS demand-paged message, causes the message to be returned to the message queue and made available for retransmission before normal input/output operations continue.
- Any function abort resets the ATTACH states to those in effect at the last sync point.

LUSTATUS - function abort might never be sent to IMS in reply to any RU indicating RQD2 or RQE2, or the session is terminated.

- LUSTATUS - Commit

  IMS does not send LUSTATUS - commit. IMS receives LUSTATUS - commit as an end-of-message indication immediately following the last page of autopaged input and with EB while awaiting conversational input. This creates a "normal" end of the conversation by invoking the Conversational Abnormal Termination exit routine with a new input vector of X'28'. This allows the exit routine to schedule a transaction to commit pending resources reflected in the scratchpad area (SPA).

- LUSTATUS - NO-OP

  IMS sends and receives LUSTATUS - NO-OP.

  IMS sends LUSTATUS - NO-OP:

  - To allow RQE1 and change-direction (CD) to flow after SIGNAL RCD.
  - To end the bracket after receiving a synchronous input message that generates no output. In IMS, this only occurs for the FORMAT, /RDISPLAY, and /DISPLAY commands when attached synchronously. The LUSTATUS - NO-OP is sent indicating RQD1 and EB.
  - To end the bracket at those times when bound in-brackets/SEND and no synchronous output or restart is possible. That is, IMS is not in conversation or response mode after session restart. The LUSTATUS is sent, indicating RQD1 and EB.
  - To allow required input when bound in-brackets/SEND at session restart and a pending conversational message is dequeued using STSN protocol. LUSTATUS is sent indicating RQE1 and CD.
  - To end the bracket after receiving exception or ERP FMH7 (without EB) sense code X'0864', following response mode or the last output message of an IMS conversation.

  LUSTATUS - NO-OP can be received during asynchronous or synchronous input processing and might indicate either CD or EB, as necessary. Receipt of LUSTATUS EB during an IMS conversation or response mode or while in test mode causes termination of the IMS conversation, response, or test mode and causes the associated output message to be dequeued. For conversation, the Conversational Abnormal Termination exit routine is invoked just as if an /EXIT command had been received.

  When IMS is the secondary half session, it receives a special case of LUSTATUS BB/EB RQ*1 after responding with DR1 to a BID command sent by the primary half session. This LUSTATUS removes IMS (secondary) from a DFC in-brackets/pending state that was set after the aforementioned DR1 was sent and the primary half session has no other message available to be sent. This condition might occur when, for example, the application within the primary half session that caused the BID to be sent

might have been abnormally terminated during the time between sending the BID and receiving the BID response.

LUSTATUS-CD should not be followed by another LUSTATUS-CD. Doing so creates a back-and-forth effect between the half sessions. IMS always responds LUSTATUS-EB when receiving LUSTATUS-CD and no output is available to be sent.

Related concepts

"Handling IMS response mode or conversational output errors" on page 505
This topic describes how IMS handles response and conversational mode errors during an ISC session and how to keep the half sessions in sync.

Paged messages ERP

For sender-detected errors (/DEQ command) detected after transmission of an IMS multichain demand or autopaged output message has been initiated, IMS must send the CANCEL command (as necessary) in conjunction with LUSTATUS - abort.

The CANCEL command is used to terminate a single multi-RU chain, or page, while the LUSTATUS is used to terminate the process receiving the multichain, or paged, message. If IMS receives a CANCEL during an input autopaged message without receiving the subsequent LUSTATUS - abort on the next input RU, IMS returns an exception response indicating selective receiver ERP followed by an FMH7 with the abort sense code X'0865' and an appropriate ERP message. Only the LUSTATUS - abort need be sent or received when the error is detected while between chains.

Errors detected on the first page of an autopaged input message result in IMS either sending a contention sense code or a selective receiver ERP sense code followed by an FMH7 indicating one of the nonfunction abort sense codes and an appropriate error message. Errors detected on nonfirst pages of an autopaged input message or for input page request received during an IMS demand-paged output message result in IMS sending the selective receiver ERP sense code followed by an FMH7 indicating the X'08650000' function abort sense code and an appropriate error message. The function abort sense code is used in these cases to cause the sending process (of autopaged input or demand-paged requests), rather than just a single chain or page, to be terminated.

Ready-to-receive protocol

IMS does not send the SNA ready-to-receive (RTR) command, but receives the command when acting as a primary half session.

When receiving the RTR command, IMS responds either with a DR1, followed by an available output message, or with an exception DR1 response (X'08190000'), if output is either not available or cannot be sent.

RSHUT protocol

IMS does not send, but does receive, the RSHUT command.

When receiving RSHUT, IMS sends a DR1 response and schedules termination of the session at the next convenient point. The session is normally terminated at the end of the current input or output chain.

Selective receiver ERP

Selective receiver error recovery procedure (ERP) is initiated using the X'08460000' sender ERP sense code.

The procedure has these characteristics:

- The procedure is symmetrical to both half sessions.
- ERP messages can be sent by the sender of the exception response indicating selective receiver ERP. An ERP message is an ERP FMH7 header followed by an IMS error message. Each selective receiver ERP message is preceded by an ERP FMH7 function management header.

- Contention for ERP messages does not occur for the exception response sender.
- ERP messages can consist of a full chain of arbitrary length, but must immediately follow the exception response indicating selective receiver ERP.
- The ERP message is recognized by the receiver for special processing.
- The response sender has an opportunity to reset the function management level bracket and send/receive states.

The ERP messages sent by IMS have associated default IMS message output descriptors (MODs) and display output formats (DOFs). The error message is formatted using SNA character string (SCS) controls even when sent to a component defined with MFS.

If necessary, the sender of the exception response enters into an ERP PURGE mode until both half sessions' send/receive states are again synchronized. The receiver of the exception response must cause the resynchronizing event.

Related concepts

SNA character string controls (System Programming APIs)

Related reference

“Error recovery procedure FM header” on page 543
IMS sends the FMH7 ERP header and its associated message following a selective receiver ERP exception response (X'0846').

“ERP purging” on page 525
After sending an exception response and before continuing to send or receive, the exception response sender might need to enter error recovery process (ERP) PURGE mode until the DFC state managers of both half sessions are synchronized.

“Sender ERP” on page 536
The SNA CANCEL request can be used during sender error recovery processing as necessary to terminate the chain in error and to provide synchronization between half sessions. CANCEL can be either solicited (by the sender of the exception response) or unsolicited (because of a receiver-detected error, which results in an exception response).

Selective receiver ERP sense codes supported

During output, IMS recognizes receipt of the SNA-defined selective receiver ERP system sense code when assuming either half-session role, returns the message to the queue, ensures that a synchronizing event occurs to end the other half session's ERP purge cycle, and enters a receive state that can be satisfied only by input (ERP message) from the other half session.

When receiving this ERP message, IMS takes one of three actions based on the FMH7 sense code and the DFC protocols associated with the received ERP message:

- The ERP message (DFS2083) is routed to the IMS master terminal operator as a warning that an error condition was recovered on the ISC session. The ISC output message in error is either dequeued or retransmitted, and normal input or output operations continue. The message is retransmitted when an FMH7 is received with sense code X'0866' and without EB. The message is dequeued when an FMH7 is received with EB during test, response, or conversational mode output.
- The ERP message (DFS2073) is routed to the IMS LTERM associated with the IMS terminal operator who was the source (using an IMS message switch) of the ISC message switch output in error. The ISC output message in error is dequeued and normal input or output operations continue.
- The ERP message (DFS2073) is routed to the IMS master terminal operator and the ISC session is terminated. The ISC output message in error remains on the IMS message queue.

  However, if ERPKPSES=Y and the sender ERP sense code is X'08460000', the original message is dequeued and the session remains active. In this case, which terminal the ERP message (DFS2073I) is passed to depends on the specification of the COMPT*n* parameter and which side initiated the message in error. If the original message is initiated from the secondary side of the session and COMPT*n*=SINGLE2 | MULT1 | MULT2, the ERP message is routed to the original inputting terminal. Otherwise, the ERP message is routed to the MTO.

All ERP messages must carry an EB or CD or the session is terminated. Further, EB received with the ERP message forces end of IMS conversation, response, or test mode. When conversation mode is terminated, the Conversation Abnormal Termination exit routine is also invoked as would occur when an IMS /EXIT command is received. IMS processes FMH7 ERP messages received based on the FMH7 sense code and the DFC protocols associated with the message as in these subtopics.

X'0864xxxx': function abort

The loop occurs upon retransmission. The sender should not resend the data.

The following table shows how IMS processes the FMH7 ERP messages with sense code X'0864xxxx'.

Table 97. IMS processing for sense code X'0864xxxx'

Message type	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Response mode			
FMH7 W/EB	X		
FMH7 W/CD"1" on page 533	X		
Non-last conversation			
FMH7 W/EB"2" on page 533	X		
FMH7 W/CD		X	
Last conversation			
FMH7 W/EB"2" on page 533	X		
FMH7 W/CD"1" on page 533, "2" on page 533	X		
Message switch			X
Test mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Other Message Types			
FMH7 W/EB	X		
FMH7 W/CD"1" on page 533	X		

Notes:

1. FMH7 received with CD schedules LUSTATUS-NO-OP (X'0006') with EB to be returned next on the session if the output in error was a response mode reply, the last conversational output, or an ATTACH without SCHEDULER.
2. Conversation Abnormal Termination exit routine is invoked as would occur for /EXIT.

X'0865xxxx': function abort

The sender is responsible for detecting the loop.

The following table shows how IMS processes the FMH7 ERP messages with sense code X'0865xxxx'.

Table 98. IMS processing for sense code X'0865xxxx'

Message type	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Response mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Non-last conversation			
FMH7 W/EB"1" on page 534	X		
FMH7 W/CD		X	
Last conversation			
FMH7 W/EB"1" on page 534	X		
FMH7 W/CD		X	
Message switch			X
Test mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Other Message Types			
FMH7 W/EB		X	
FMH7 W/CD		X	

Notes:

1. Conversation Abnormal Termination exit routine is invoked as would occur for /EXIT.

Related concepts

SNA character string controls (System Programming APIs)

X'0866xxxx': function abort

The receiver is responsible for detecting the loop.

The following table shows how IMS processes the FMH7 ERP messages with sense code X'0866xxxx'.

Table 99. IMS processing for sense code X'0866xxxx'

Message type	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Response mode			
FMH7 W/EB	X		
FMH7 W/CD	X		
Non-last conversation			
FMH7 W/EB"2" on page 535	X		
FMH7 W/CD	X		
Last conversation			

Table 99. IMS processing for sense code X'0866xxxx' (continued)

Message type	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
FMH7 W/EB"2" on page 535	X		
FMH7 W/CD	X		
Message switch	X		
Test mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Other Message Types			
FMH7 W/EB	X		
FMH7 W/CD"1" on page 535	X		

Notes:

1. FMH7 received with CD schedules LUSTATUS-NO-OP (X'0006') with EB to be returned next on the session if the output in error was a response mode reply, the last conversational output, or an ATTACH without SCHEDULER.
2. Conversation Abnormal Termination exit routine is invoked as would occur for /EXIT.

During input, IMS can detect many types of input errors and internal processing conditions, such as undefined transaction codes, incorrect transaction formats, security violations, SNA protocol and data structure errors, and sync-point request errors. These errors can occur on almost any SNA request element of the message and might result in IMS sending an exception response indicating a selective receiver ERP sense code; purging the input message until a synchronizing CANCEL command, RQD1 or RQD2, change-direction indicator, or CHASE command is received; and then sending a selective receiver ERP message.

IMS provides the following sense information in the ERP FMH7 sent with the ERP message:

X'08260000'
Used for input and internal processing errors, other than the errors described in the remainder of this list, that are detected where a normal IMS ERP message is sent.

X'10030000'
Sent when the ATTACH ATTPRN value is not known to IMS.

X'08650000'
A function abort sense code for which the data sender is responsible for detecting and preventing loops. Function abort X'08650000' is sent by IMS after receiving an unsolicited CANCEL command (sender-detected error) during an MFS-autopaged input message that does not result in a DFC between-brackets state. Function abort X'08650000' is also sent by IMS for errors detected on input page requests received during an IMS demand-page output message and for errors detected on nonfirst pages of an autopaged input message.

X'084B0000'
Sent when the attach ATTDPN value is not available for the component entering the data. That is, the input DPN value was not ISCEDT, the optional ISC edit alias, or basic edit; or MFS was not available for the component. However, if MFS was available but the DPN value was not a valid MID, sense code X'08260000' occurs rather than X'084B0000'.

X'080F0000'
Sent when the ATTACH data stream profile (DSP) value does not define a valid or defined component number for the half-session name within IMS.

X'1008xxxx'
The user field (xxxx) is defined by SNA as follows:

X'6001': Invalid ATTACH FM header ATTDBA value
X'1204': Invalid version ID on FMH4

Other types of errors detected during input or output operations that cause immediate session termination are:

- Bracket protocol violations
- Send/receive protocol violations
- Unsupported response requests
- Unsupported or invalid FMH types and formats
- Other major VTAM-detected errors or conditions such as LOSTERM, buffer pool or copy RPL space exhausted, or input RU truncation.

Related reference
“Paged messages ERP” on page 531
For sender-detected errors (/DEQ command) detected after transmission of an IMS multichain demand or autopaged output message has been initiated, IMS must send the CANCEL command (as necessary) in conjunction with LUSTATUS - abort.

Receiver-detected errors during data flow reset state

The following sense codes can be sent and received when the session flow is not in data flow active state, as, for example, during bind, resynchronization (STSN), or start data traffic (SDT).

X'0845xxxx'
Bind rejected because of invalid user data.

X'0847xxxx'
Restart mode mismatch detected by secondary half session at receipt of BIND1 STSN or SDT.

X'08210000'
Bind rejected because of invalid session parameters.

X'080Dxxxx'
Bind rejected because of bind race.

Related concepts
“Resolving a bind race” on page 491
A race occurs when IMS and another logical unit simultaneously send BIND requests to each other and the two half-session names are mirror images.

Sender ERP

The SNA CANCEL request can be used during sender error recovery processing as necessary to terminate the chain in error and to provide synchronization between half sessions. CANCEL can be either solicited (by the sender of the exception response) or unsolicited (because of a receiver-detected error, which results in an exception response).

If necessary during chain RU or autopaged output, the sender of the exception response enters into an ERP PURGE mode until both half sessions' send/receive states are again synchronized. The receiver of the exception response must cause the resynchronizing event. This event can be one of the following:

- An RQD1 or RQD2
- Receipt of change-direction
- A CANCEL request (caused by either RQD1 or RQE1/CD), if the exception response occurs during an output chain
- A CHASE request (because of RQD1 on CHASE), if the exception response is received while IMS is between RQE1 output chains (pages) of an autopaged output message

Related reading: For more information on sender error recovery procedure (ERP), see *z/OS Communications Server: SNA Programming*.

Sender ERP sense codes

The only sender ERP sense codes sent between two IMS subsystems are X'08130000' and X'0846xxxx'.

The valid contention sense codes received by IMS and additional operations performed are:

- X'08130000'—Bracket reject; no RTR sent

  No ready-to-receive (RTR) condition is set. When the pseudo-wait is satisfied, the message is retransmitted from the beginning. This sense code can also be used to reject the SNA BIS command sent by IMS.

- X'08140000'—Bracket reject; RTR sent

  The message is returned to the queue, and an RTR pending state is entered. No output is sent while IMS is between-brackets until an RTR is received. However, messages flow, subject to the bracket and send/receive protocol defined for the message type or component, when IMS is again left in an in-brackets/SEND state following receipt of change-direction.

After either of these sense codes, IMS attempts to send the output message.

IMS recognizes receipt of contention system sense code X'08130000' when assuming a primary half-session role, returns the output message to the message queue, and enters a pseudo-receive state that can be satisfied by:

- Receiving input from the secondary half session
- Being posted for output as a result of an IMS master terminal operator command, a message switch from another logical unit, or additional messages inserted by an IMS application program

After either of these actions, IMS attempts to send the output message.

IMS only indicates contention when assuming a secondary half-session role and receiving either input data or normal flow commands (BID and BIS) after having already initiated a bracket to the primary half session. IMS sends the X'08130000' sense code to indicate contention in these cases. The X'08140000' contention sense code is not sent by IMS.

Additional exception response sender ERP sense codes sent and received are:

- X'08190000'—No output available

  This code is sent by IMS when assuming the primary half-session role and no output is immediately available following receipt of a ready-to-receive (RTR) indicator. IMS does not receive this sense code because IMS does not send RTR.

- X'0846xxxx'—Selective Receiver ERP

  IMS sends and receives the SNA selective receiver ERP sense code. IMS sends this sense code when a response mode, conversational transaction, or application abnormally terminates. xxxx is a user sense field that is ignored by IMS when it is received. However, IMS does include this sense code in the message that is sent to the master terminal or to the terminal operator that was the source of the message. During output, it is normally set to the binary value of the IMS error message number that is sent as the ERP message when sending the response. However, the user sense code can be set to X'0000' as, for example, when the error, such as an application abend, occurs for a response mode or conversational transaction.

- X'0864xxxx'—Function abort. Loop occurs upon retransmission. Sender should not resend data.

  IMS receives, but does not send, this function abort sense code. When receiving this code, IMS dequeues the associated output message (if still active) and then continues with normal input or output operations. xxxx is a user sense field ignored by IMS. An LUSTATUS - abort is sent by IMS if input is received prior to the end of an IMS MFS-paged output message.

  The function abort sense code X'0864xxxx' cannot be sent to IMS during nonlast IMS conversational output; otherwise, the output message is returned to the queue. The master terminal operator is

notified and the session is terminated. When the function abort sense code X'0864xxxx' is sent following the last conversational output message, the message is dequeued and the Conversational Abnormal Termination exit routine is invoked just as occurs for /EXIT.

- X'0865xxxx'—Function abort. Sender responsible to detect loop.

  IMS receives, but does not send, the sender ERP abort sense code. If received, this code causes IMS to return a message to the queue and close the session. IMS retransmits the message from the beginning at the next opportunity after session restart. xxxx is a user sense field ignored by IMS.

- X'0866xxxx'—Function abort. Receiver responsible to detect loop.

  This sense code causes IMS to retransmit the message from the beginning at the next opportunity. The session is not terminated. xxxx is a user sense field ignored by IMS.

- X'08670000'—Multichain ERP purge.

  IMS sends the multichain purge sense code at the end of ERP purging. IMS receives the multichain ERP purge sense code only after receiving an exception response to a nonfirst or nonlast page of an MFS output autopaged message. Non-MFS autopaged output and the first and last page of an MFS output autopaged message are sent requesting definite response. Exception to these SNA chains can result only in single chain purge.

Other sender ERP sense codes cause IMS to notify the master terminal operator and terminate the session.

Related reference

“Selective receiver ERP” on page 531
Selective receiver error recovery procedure (ERP) is initiated using the X'08460000' sender ERP sense code.

“ERP purging” on page 525
After sending an exception response and before continuing to send or receive, the exception response sender might need to enter error recovery process (ERP) PURGE mode until the DFC state managers of both half sessions are synchronized.

Sender-detected errors on nonpaged messages

When an error is detected after initiating transmission of a single SNA nonpaged, multi-RU message, the sender can send the SNA unsolicited **CANCEL** command.

IMS sends unsolicited **CANCEL** only if a current output message is terminated by an operator **/DEQUEUE** command entered prior to the last segment of the message (single SNA chain) being transmitted.

If IMS receives an unsolicited **CANCEL** while data is being received by IMS, the message is discarded or backed out.

Sense codes that IMS receives

A *sense code* is a 2-byte field containing the category and modifier defined for the particular exception condition that has occurred. The 2 bytes following the sense code can contain optional user data, and are not supported for all sense codes.

The following table shows the input sense codes that IMS receives.

Table 100. Sense codes that IMS receives

Input sense code name	Input sense code	Sender ERP	FMH7	LUSTATUS
Commit/NO-OP	- X'00060000'			X
Queue empty	- X'00070000'			X
Bind race	- X'080D0000'	X		
Contention	- X'08130000'	X		

Table 100. Sense codes that IMS receives (continued)

Input sense code name	Input sense code	Sender ERP	FMH7	LUSTATUS
Contention	- X'08140000'	X		
Invalid parms	- X'08210000'	X		
Sel Rcvr ERP	- X'08460000'	X		
Reject restart	- X'08470000'	X		
Abort	- X'08640000'	X	X	X
Abort	- X'08650000'	X	X	X
Abort	- X'08660000'	X	X	X
ERP purge	- X'0867'	X		
Other		Note 1	Note 1	Note 1

Notes:

1. Other sense codes cause IMS to notify the master terminal operator and terminate the session.

Related reading: For more information on sense codes, see *z/OS Communications Server IP and SNA Codes*.

Sense codes that IMS sends

The following table shows the output sense codes that IMS sends.

Table 101. Sense codes that IMS sends

Output sense code name	Output sense code	Sender ERP	FMH7	LUSTATUS
NO-OP	- X'00060000'			X
Queue empty	- X'00070000'			X
Bind race	- X'080D0000'	X		
ATTDSP	- X'080F0000'		X	
Contention	- X'08130000'	X		
No output	- X'08190000'	X		
Invalid parms	- X'08210000'	X		
Not supported	- X'08260000'		X	
Sel Rcvr ERP	- X'08460000'	X		
Reject restart	- X'08470000'	X		
ATTDPN	- X'084B0000'		X	
Abort	- X'08650000'	X		X
ERP purge	- X'0867'	X		
ATTPRN	- X'10030000'		X	
FMH4 Version ID	- X'10081204'		X	
ATTDBA	- X'10086001'		X	

SIGNAL protocol

IMS sends the SIGNAL command to request change-direction (SIGNAL RCD - X'00010000') at the end of an input message chain that does not indicate change-direction and that produces output that must be sent prior to subsequent input.

For example, IMS replies to input commands, response mode transactions, conversational transactions, or input while in test mode. SIGNAL RCD is sent before IMS sends any required DR1 or DR2 response for the input message.

Based on the IMS response protocol and the point at which IMS sends SIGNAL RCD, IMS requires that the next input following SIGNAL RCD be either LUSTATUS or CHASE indicating CD or EB. If neither of these is received by IMS, the session is terminated. Change-direction allows IMS to send the pending output message; end-bracket causes the message to be dequeued and the conversation mode, response mode, or test mode to be terminated. No other input can be processed until the pending output message has been successfully transmitted or dequeued.

After receiving SIGNAL RCD while in bracket state, IMS sends change-direction either at the end of the current output message or immediately on receipt of an LUSTATUS. A SIGNAL RCD received while between brackets prevents IMS from sending further output messages until normal flow input (data or SNA command) occurs.

Related reference

“Signal protocol example” on page 651
The following example illustrates the use of SIGNAL RCD.

Symmetrical session shutdown for LU 6.1 (SBI and BIS)

Two data flow control commands allow a symmetrical and orderly termination for peer level LU 6.1 half sessions: stop bracket initiation (SBI) and bracket initiation stopped (BIS).

SBI is a VTAM-expedited flow command, and BIS is a normal flow command.

These commands control only the normal flow requests that initiate new brackets. They do not preclude replies that can flow within existing brackets. Either half session can send SBI at any time to request the receiving half session to suppress future initiation of a bracket. After the receiver of the SBI reaches a point between brackets that is acceptable for shutdown, the receiver returns the BIS command to the SBI initiator. The BIS sender then enters a "no begin-bracket" or "NOBB" state, during which that half session can receive bracket initiation requests and may send requests or replies in the normal flow when in-brackets, but cannot initiate a bracket on its own. The primary half session is responsible for detecting and resolving SBI race conditions by maintaining indications of which half session is in NOBB state and terminating the session if both half sessions reach this state.

A successful completion of a shutdown sequence refers to successful entry into the NOBB state by both half sessions— that is, normal session termination.

IMS sends the SBI command either when the IMS /QUIESCE command is entered or when the quiesce option is specified on the /CHECKPOINT command. When BIS is returned from the other subsystem, IMS continues with the QUIESCE or CHECKPOINT command process. If the /QUIESCE command was entered and a CHECKPOINT QUIESCE is not in progress, IMS sends BIS at the next transition to a between-brackets state. If a CHECKPOINT QUIESCE is in progress when IMS receives BIS, IMS continues to send output according to other /CHECKPOINT parameters (FREEZE, PURGE, and DUMPQ) and then sends BIS when the shutdown process is complete.

After receiving an SBI or BIS, IMS sends BIS at the next between-brackets state when a CHECKPOINT QUIESCE is not in progress, or at the end of the shutdown process when a CHECKPOINT QUIESCE is in progress.

When IMS is the primary half session, IMS checks after either sending or receiving BIS to ensure that both half sessions are in NOBB state. If both are, IMS resets all message counts and terminates the session. Otherwise, IMS continues with normal processing. When IMS is the secondary half session, IMS waits for the primary half session to terminate the session. If an LTERM subpool is allocated to the session, it is deallocated (and associated message counts are reset) prior to the session termination when

both half sessions are in NOBB state. This deallocation/termination process allows a subsequent cold start of the session and frees any allocated subpools for use by other sessions. This process also ensures that no resynchronization or recovery is required when the session is restarted.

Because initiating a bracket is not allowed after the NOBB state is entered, a problem occurs when a single-bracket-chain (begin- and end-bracket) input message requires an immediate reply (such as a system or error message) that initiates a bracket. Rather than preclude end-bracket on these input messages while in NOBB state, IMS accepts and continues to process all input. When a reply occurs that would require IMS to initiate a bracket while in NOBB state, the session is terminated.

While IMS is in response mode, in conversational mode, or waiting for a Fast Path response, subpool deallocation cannot occur nor can BIS be sent, because no EB has flowed either on output or input (that is, IMS is not between-brackets).

Related reference

"SBI/BIS examples" on page 649
IMS can be either the primary half session (PHS) or the secondary half session (SHS) for all of the examples given. Therefore, all of the functions and commands shown for PHS and SHS are IMS functions and commands.

Function management headers

In SNA, function management (FM) headers are an optional part of the request unit sent over a link. This topic describes the FM headers supported by IMS on ISC sessions.

The headers addressed in this topic include:

- The ATTACH function management header. This type 5 header is used to attach a process so it can receive session input. It carries the name of the process to be attached synchronously, as well as other parameters to be used by the attached process. All messages have an explicit or implicit ATTACH FM header. The header is implicit when not actually sent with the message. An implicit ATTACH FM header assumes reset parameter values or values of a previous ATTACH as defined later in this topic.
- The error recovery procedure (ERP) function management header. This type 7 header is sent after an error condition requiring an error message to be sent is detected.
- The reset attached process (RAP) function management header. This type 5 header is input to IMS to cause the attached process and the associated data flow control states to be reset.

The following header types are subordinate to the ATTACH FM header and cannot be used unless the ATTACH FM header has previously been sent within that bracket to attach the associated process:

- The SCHEDULER function management header. This type 6 header is sent after the SCHEDULER process has been attached. It carries the name of the process to be asynchronously scheduled within IMS, as well as other parameters required by the SCHEDULER process.
- The SYSMSG function management headers. The type 6 system message headers precede system messages sent by or received by IMS.
- The data descriptor function management header. This type 4 header is used in conjunction with MFS to send a data structure name, version ID, or a current output field tab separator.
- The QMODEL function management headers. These type 6 headers are used for demand-paged messages sent by IMS Message Format Service.

All function management headers must be on a first-in-chain or only-in-chain. The following table summarizes the header types that IMS sends and receives (Y=Supported). These header types are described in the topics that follow.

Table 102. Function management header types

Function management header type	Sent by IMS	Received by IMS
FMH4 - Data Descriptor	Y	Y
FMH5		

Table 102. Function management header types (continued)

Function management header type	Sent by IMS	Received by IMS
ATTACH	Y	Y
Reset Attach Process (RAP)		Y
FMH6 - QMODEL		
QGET		Y
QGETN		Y
QPURGE		Y
QSTAT	Y	
QXFR	Y	
FMH6 - SCHEDULER	Y	Y
FMH6 - SYSMSG		
SYSSTAT (default)		Y
SYSSTAT	Y	Y
SYSERROR	Y	Y
FMH7 - ERP	Y	Y

When receiving a header, IMS ignores any coded parameters beyond the last parameter to which it is sensitive (that is, that IMS supports). The other LU 6.1 subsystem should do the same when receiving headers that IMS sends.

Using FM headers to invoke ISC edit

In the following situations, IMS can use ISC edit (ISCEDT) to edit transactions, commands, and message switches between LTERMs for an IMS-ISC session

- The attach manager is in the reset state (between-brackets or following a RAP), and either no ATTACH FM header or an ATTACH FM header that does not specify a destination process name (ATTDPN) is received.
- The destination process name (ATTDPN/SCDDPN) within the ATTACH or SCHEDULER FM header equals ISCEDT or indicates the user-defined alias for ISC edit as defined during system definition on the COMM macro statement.
- The attach manager is not in the reset state, the active process is ISC edit, and either no ATTACH FM header or a header that does not specify an ATTDPN is received.

Input that does not specify the ATTPRN/SCDPRN parameter is edited only for the transaction code and password at the beginning of the message. The optional IMS password and leading characters less than X'41' are deleted during editing. The format and requirements for the IMS transaction code and password are the same as for basic edit. No editing of any kind occurs for the remainder of the message.

Input parameters passed from the attach manager or SCHEDULER (if supplied) are return destination process name (ATTRDPN/SCDRDPN), return primary resource name (ATTRPRN/SCDRPRN), and primary resource name (ATTPRN/SCDPRN).

The input ATTPRN/SCDPRN parameter is passed from the attach manager or SCHEDULER as an IMS transaction code or as the LTERM name on a message switch, as defined under “Initiating a process: ATTACH FM header” on page 543. When the ATTPRN/SCDPRN parameter is supplied, no editing of any kind occurs for the input. ATTPRN/SCDPRN defines an IMS destination, but does not provide an input password to IMS transaction security (if this security is defined for the ISC node).

Related concepts

"Basic edit" on page 434
If you do not use MFS, an IMS function called basic edit performs message editing.

Related reference

"Examples using ISC edit ATTACH parameters" on page 577
The following topics provide examples of using ISC edit ATTACH parameters.

Initiating a process: ATTACH FM header

The ATTACH FM header is used to attach a process to receive session input. It carries the name of the process to be attached synchronously, as well as other parameters used by the attached process.

Because ATTACH is defined for synchronous execution, you must understand the relationships between the process to be attached, the IMS message types and execution modes, and the SCHEDULER process.

IMS sends the ATTACH FM header without a SCHEDULER FM header:

- On the conversational, response mode, or IMS command output message replies resulting from an input ATTACH.
- On asynchronous replies that result from a message received on the same session with ATTACH EB.
- When the other half session is bound without SCHEDULER model support.
- On system messages sent by IMS through SYSMSG.

An IMS message can consist of either single or multiple SNA chains as indicated within the ATTACH ATTIU parameter. The ATTACH FM header can be present only once for each input or output IMS message. For output MFS demand-paged messages, the ATTACH FM header is present only on the first output SNA chain. The first input paging request to the demand-paged output must contain an ATTACH (for DPN=QMODEL); subsequent input page requests for the same demand-paged output message can optionally contain an ATTACH (which must also indicate DPN=QMODEL).

Related concepts

"Relationship of ISC and IMS execution modes" on page 461
Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

Related reference

"ATTACH FM header format" on page 557
The format of the ATTACH FM header is defined in the following table.

Error recovery procedure FM header

IMS sends the FMH7 ERP header and its associated message following a selective receiver ERP exception response (X'0846').

An ERP exception response need not be received by IMS before receiving an FMH7 ERP header and associated message. FMH7 messages sent by IMS indicate either RQE1/CD or RQD1/EB, as appropriate. ERP messages that are sent or received by IMS are limited to a single SNA chain of SCS characters and must not contain VLVB format records. FMH7 messages received by IMS must indicate either CD or EB. RQE1 or RQD1 is allowed on either.

Single-segment error messages that are created within IMS are 79 characters or less in length. Multi-segment messages that are created within IMS can be greater than 79 characters in length, but each segment has a 79-character maximum length.

Error messages that are created by input to the ISC error message process for output to the master terminal or source operator terminal create single- and multi-segment output messages (DFS2073 or DFS2083).

IMS inserts error messages into IMS message queues by using a default MFS MOD name. These messages are formatted as defined by the MOD during output to a component defined with MFS DPM. These MODs and associated DOFs are contained in the MFS format library.

Each selective receiver ERP message is preceded by an ERP message header that includes the specific system and user error sense codes.

Related reference

“Selective receiver ERP” on page 531
Selective receiver error recovery procedure (ERP) is initiated using the X'08460000' sender ERP sense code.

“Error recovery procedure (ERP) FM header” on page 566
The following table shows the format of the ERP FM header.

Resetting the active process: RAP FM header

The reset attached process (RAP) FM header is used to reset the receiving half session's active process and all session states, except bracket and send/receive states, to the equivalent of a between-brackets state.

IMS receives the RAP FM header, but does not send it. After receiving the RAP request, IMS responds with an ATTACH SCHEDULER, or with LUSTATUS - queue empty if no output is available to be sent. If the RAP request is received during an IMS demand-paged output message, the message is dequeued (committed) before a check for more output is made. The IMS operation performed for the RAP request is the same as for the NEXTMSG operator control request provided with MFS input or the PA2 request from a 3270 device.

The RAP request must be sent without data and must indicate change-direction, RQD2, or RQE2.

The following figure illustrates the use of the RAP FM header.

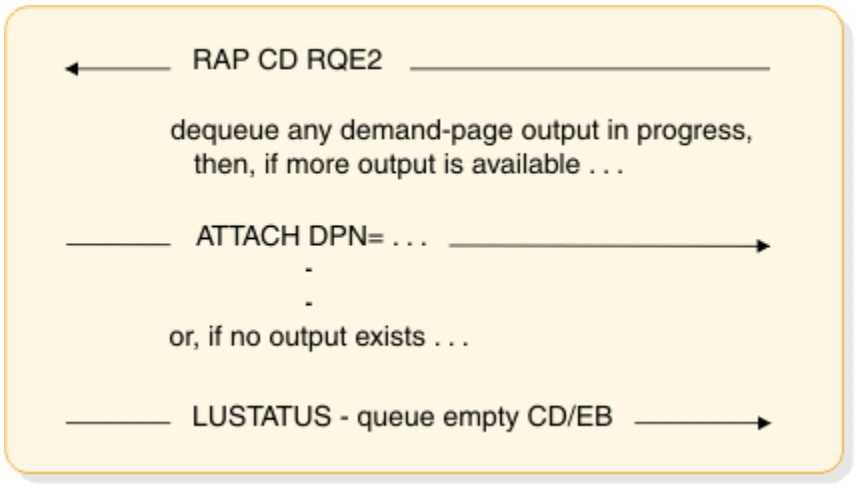


Figure 71. RAP FM header example

Related reference

“Reset attached process (RAP) FM header format” on page 573
The following table shows the format of the Reset Attach Process (RAP) FM header.

Requesting asynchronous transaction processing: SCHEDULER FM header

ATTACH is defined for synchronous scheduling and execution; SCHEDULER is defined for asynchronous scheduling and execution.

The SCHEDULER process can be attached to send messages to be processed asynchronously. That is, IMS perceives the messages as being processed without regard to scheduling or execution timing, whether output will result at all, and ignoring any synchronous relationship between the input and any resulting reply. Each half session between the logical units might indicate, by using the session bind parameters, whether the SCHEDULER process is supported and available to receive asynchronous messages. The IMS half session always indicates the SCHEDULER process is available. The other half session might indicate that the SCHEDULER process is or is not available to receive asynchronous input.

When the bind indicates that the other half session does not support the SCHEDULER process, IMS defaults to sending each output with the ATTACH FM header.

The SCHEDULER process is attached by indicating the SNA name (X'02') as the ATTACH DPN parameter and by concatenating the SCHEDULER FM header to the ATTACH FM header. During the time that the SCHEDULER is attached, each subsequent message to be scheduled asynchronously must be sent with at least the SCHEDULER FM header, but does not require another ATTACH FM header. Variable-length ATTACH parameters passed to the SCHEDULER are ignored on input and are not sent with output by IMS. The SCHEDULER FM header carries the name of the process to be scheduled (SCDDPN), as well as information necessary for scheduling and return-reply routing. SCDDPN is a required parameter for the SCHEDULER and must be supplied on IMS SCHEDULER input. SCDDPN on messages sent by IMS are set from a previous input SCDRDPN or optionally by MFS using the output message format descriptor. When no other value is available, SCDDPN is set to "ISCEDT" as a default.

The SCHEDULER process is always available for IMS input. IMS sends an ATTACH with the ATTDPN parameter set to X'02' (SCHEDULER) concatenated with a SCHEDULER FM header under each of the following conditions:

- When the other half session supports the SCHEDULER model (as defined in the bind parameters).
- When sending an asynchronous reply resulting from input entered previously on the session and scheduled using the SCHEDULER process.
- When sending unsolicited asynchronous messages. Examples of such messages are an IMS message switch or an output reply resulting from an application not scheduled directly as a result of input on the session.

Within IMS, the SCHEDULER supports both single- and multiple-chain input and output messages using the ATTACH ATTIU parameter. IMS does not permit multiple input or output messages to be scheduled using the ATTACH multiple-chain indicator.

The SCHEDULER FM header can be present only once for each input or output IMS message. For output MFS demand-paged messages, the SCHEDULER FM header is present only on the first output SNA chain (first output MFS page). IMS does not send or receive the SCHEDULER SCHEDSTAT, PURGE, or PURGSTAT FM headers.

Related reference

“Data flow control protocol reference” on page 515
The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

“SCHEDULER FM header format” on page 573
The following table shows the format of the SCHEDULER FM header.

System message process (SYSMSG) and related FM headers

The system message process is indicated by a process name in the ATTACH FM header.

Input system messages are routed to the master terminal operator when no ATTPRN is supplied in the ATTACH or SYSERROR FM header for a SYSMSG. This routing is accomplished by converting the incoming SYSMSG into IMS system message DFS2072. If an ATTPRN value is supplied, it becomes the IMS input message destination transaction code or LTERM name. The system message process cannot be used to access the IMS command processor.

A reply can result from an input SYSMSG if the ATTPRN value supplied is for an IMS transaction. This reply is returned through ISC edit or MFS, depending upon the output component definition.

IMS inserts system messages into IMS message queues using a default MFS MOD name. These messages are formatted as defined by the MOD during output to a component defined with MFS DPM. These MODs and associated DOFs are contained in the MFS format library.

IMS sends SYSMSG only for:

- IMS broadcast messages
- System messages that IMS sends after sending a response to input. These system messages can occur instead of the application sending a reply (for example, when an abnormal termination occurs). IMS system messages directly solicited by an IMS command are sent as normal replies. Other system-generated messages can result from error conditions. These messages are converted to exception responses to the input transaction and are sent as error messages using the error recovery process (ERP).

MFS might edit an output SYSMSG, depending on the output component definition and whether the reply was inserted with an MFS MOD name.

Messages sent to or received by SYSMSG can include either the SYSSTAT or SYSERROR FM header. The SYSSTAT FM header is assumed by the receiver if no header is supplied by the sender. IMS can receive either header, and attaches a prefix to each header with IMS message number DFS2072 (if the message is to be sent to a master terminal operator). IMS sends broadcast output using the SYSSTAT header. All other IMS system messages sent as SYSMSG use the SYSERROR header. If the ATTACH ATTRDPN or ATTRPRN parameters are supplied on an input message that results in SYSMSG output, these parameters are included in the output SYSERROR FM header to be used by the receiver's SYSMSG process. The ATTACH ATTRDPN and ATTRPRN parameters are not sent for an IMS output SYSMSG.

Single-segment system messages created within IMS are 79 characters or less in length. Multisegment messages created within IMS can be greater than 79 characters in length, but each segment has a 79-character maximum length.

System messages created by input to the ISC SYSMSG process for output to the master terminal create single- and multisegment output messages. System messages created by input to the ISC SYSMSG process in which ATTPRN and ATTDPN are specified must have a MID capable of handling multisegment messages with a minimum of five segments.

Related reference

“SYSMSG process headers” on page 575
The following tables show the formats of the SYSMSG process headers: the SYSERROR FM header and the SYSSTAT FM header.

Chapter 31. Using MFS with ISC

Message formatting for MFS DPM is specified on the IMS system definition TERMINAL macro statement by the parameter DPM-Xn, where X might be A or B. For ISC nodes, the form DPM-Bn is always used.

This topic addresses the following:

- How MFS DPM-Bn can be invoked to format IMS input and output
- The way in which the ATTACH, SCHEDULER, and reset attached process (RAP) function management headers are used by MFS
- MFS support for the SNA-defined QMODEL and data descriptor function management headers (which control MFS processing)

Restriction: MFS is not supported for ISC TCP/IP links.

Some input MFS format errors (such as invalid cursor, incorrect output formats, or no output formats) are not detectable within IMS before IMS sends the requested response to the input message. In these cases, IMS sends the error message by using ATTACH ATTDPN=SYSMSG. This ATTACH results even if the input message indicated change-direction and exception response and is an implicit acknowledgment to the message.

An ATTACH SYSMSG received by IMS during demand-paged output is treated as an invalid MFS paging request and causes session termination. The received SYSMSG is discarded.

Related reference

“FM header format reference” on page 557
Each header's length is defined by a 1-byte length field. The value in this field includes the length field itself. A concatenation flag indicates whether an additional header follows.

Activating MFS input formatting

When MFS is used, input messages can be processed by the message and format descriptors.

When IMS receives an input message from an ISC logical unit, basic edit or ISC edit is performed unless MFS is defined for the input component (ATTDSP) and a MID name accompanies the message. The MID name can be supplied by including it either as the ATTDPN parameter on the ATTACH function management header or as the SCDDPN parameter on the SCHEDULER FM header.

The MFS escape characters (//) are not supported for ISC. When the MID name is present, MFS edits the message using the specified MID and its associated device input format (DIF).

Activating MFS output formatting for ISC

MFS output formatting occurs when an output message has an associated message output descriptor (MOD).

The MOD is supplied in one of the following ways:

- The application program supplies a MOD name with the output message.
- The input message is processed by a message input descriptor (MID) whose definition specifies a MOD name for output formatting.
- The output message is a message switch that is created by a MID whose definition specifies a MOD name for output editing.

If no MOD is associated with the output message, no MFS editing occurs. Values for the ATTACH and SCHEDULER parameters are the default values that are defined under these respective header types.

You can use the MOD to specify the next MID name to be used to format input as the result of output. In an ISC session, IMS cannot control the relationship between formatted IMS output messages

© Copyright IBM Corp. 1974, 2020

and subsequent input messages. Therefore, involvement is required by the remote subsystem or user application program to ensure that the proper MID name is used for input MFS formatting. The ATTACH and SCHEDULER RDPN/DPN parameters perform the ISC function of externalizing the next MID on output and invoking MFS formatting on subsequent input.

Related concepts

Relationships between MFS control blocks (Application Programming APIs)

Related reference

“ATTDPN” on page 561

The destination process name (ATTDPN) parameter identifies, explicitly or implicitly, the input process to be attached to the half session.

MFS Distributed Presentation Management (DPM) messages

An IMS MFS DPM message can consist of either single or multiple SNA chains as indicated within the ATTACH parameters.

For MFS DPM input or output using a device format defined with the paging OPTIONS=MSG, IMS sends or receives the entire message as a single chain of one or more related transmissions. For MFS DPM output using a device format defined with paging OPTIONS=DPAGE or PPAGE, IMS sends each logical or presentation page as a single chain of one or more related transmissions.

MFS output DPM format definitions also allow options for paged output (OPTIONS=PPAGE or DPAGE) to be sent as demand-paged or autopaged. Demand-paged output requires paging requests to the other half session for each output page. Autopaged output is sent as a series of consecutive pages (chains) with no paging requests to the other half session.

MFS DPM input using a device format defined with the paging OPTIONS=DPAGE allows IMS to optionally receive each logical page as a single chain of one or more related transmissions. All chains of autopaged input (multiple logical pages as multiple related input chains) are received consecutively without paging requests by IMS.

FM headers can occur as part of the first or only transmission of each input or output chain of the message.

MFS page delete function

The MFS page delete function is not available for use by an ISC session.

OPTIONS=NPGDEL is forced on the IMS system definition TERMINAL macro statement or an ETO user descriptor, because paging requests other than QMODEL-architected paging requests are not supported. This includes MFS operator control table requests and other forms of input data that might occur during output demand-paging. An error (exception response and appropriate ERP message) occurs if data other than a valid QMODEL paging request is received during demand-paged output. The output message is returned to the queue as a result of the error condition and retransmitted at the next opportunity. If the output demand-paged message is conversational or response mode output, it is immediately resent and input is not allowed until the output message is successfully transmitted or dequeued by a valid QMODEL paging request or appropriate ERP action.

MFS online error detection

This topic describes how IMS detects input errors, output errors, and paging errors for MFS.

Output errors

MFS output errors are detected after IMS has already sent the response to any preceding input. If an error is detected during MFS MOD or DOF block selection, an error message is sent as ATTACH SYSMSG, and the IMS message is returned to the message queue for retransmission. If the MFS test mode is in effect, it is reset.

Invalid page requests cause error messages to be sent.

Input errors

An error message is sent and the input message is rejected if one of the following occurs:

- An error is detected during MFS MID or DIF block selection.
- A nonzero version ID is received in the data descriptor FM header and it does not match the version ID in the MFS descriptor.
- An error is detected during DPAGE selection (that is, no condition is satisfied on matching the DPAGE label with the DSN in the DD FM header or the COND= with the data).

If one of the following errors is detected during multiple DPAGE input, an error message is sent to the other subsystem, and the input message is rejected:

- Multiple transmission chains. More data is present in the chain than defined for the DPAGE selected.
- Any mapped-input LPAGE contains no data segments (as a result of segment routines canceling all segments, for example).

If one of the following errors is detected during a single DPAGE input (that is, multiple DPAGE input is not requested in MFS definitions), an error message is sent and the input message is rejected:

- A single transmission chain is received and contains more data than defined for the DPAGE selected.
- Multiple transmission chains are transmitted.
- The mapped-input message contains no data segments (as a result of segment routines canceling all segments, for example).

If the input message is canceled by the User Segment Edit exit routine, or if a User Segment Edit exit routine failure is detected, an error message is sent to the other subsystem. In the latter case, the input message is rejected.

Paging errors

An error message is sent to the other subsystem when an invalid paging request (QMODEL FM header) is detected:

- The QNAME parameter on the QGETN or QGET function management header does not match the ATTDQN/SCDDQN parameter on the ATTACH or SCHEDULER FM header sent by IMS. An attached demand-paged output message is returned to the message queue for retransmission. A scheduled demand-paged output message continues to wait for a QGETN with the proper QNAME specified.
- The QORG parameter on the QGETN or QGET FM header is invalid. The output message is returned to the message queue for retransmission.
- The QGETN FM header is received and no output message is in progress.
- A paging request other than QGETN is received as the first input following scheduled demand-paged output (ATTACH SCHEDULER), or a QSTATUS FM header is sent by IMS because of an invalid cursor on a paging request.
- The QGETN FM header is received for an OLP demand-paged output and current cursor position is at the last page.

The QGET FM header causes a QSTATUS or an error message to be sent in the following cases:

- Between messages.
- Non-OLP demand-paged output message is in progress. An error message is sent and the output message is placed on the queue for retransmission.
- Cursor value does not contain a valid 2-byte binary number. An error message is sent, and the output message is placed on the message queue for retransmission.

- Cursor value is outside the range for the output message.

Related reference

"QSTATUS FM header" on page 554
A QSTATUS FM header is sent by IMS in reply to an input QPURGE FM header or because an invalid cursor was detected on a page request for a demand-paged output.

"Error recovery procedure FM header" on page 543
IMS sends the FMH7 ERP header and its associated message following a selective receiver ERP exception response (X'0846').

The ATTACH and SCHEDULER FM headers under MFS

All input and output messages include an implicit or explicit ATTACH FM header and, optionally, a SCHEDULER FM header.

MFS DPM provides format definition options to permit the insertion of input routing parameters (ATTRDPN/SCDRDPN, ATTRPRN/SCDRPRN, and ATTPRN/SCDPRN) into the input data stream. The input ATTDPN/SCDDPN is the MFS MID used to format the input message. MFS DPM provides output format definition options to permit the insertion of user-defined information into these headers.

The ATTACH FM header and SCHEDULER FM header, if required, are sent without data as the only element of the first transmission chain of demand-paged output. The first input paging request to the demand-paged output must contain an ATTACH (for DPN=QMODEL); subsequent input page requests for the same demand-paged output message can optionally contain an ATTACH (and must also indicate DPN=QMODEL). For all other MFS input and output messages the ATTACH (and SCHEDULER, if required) FM headers are sent with data as the first or only element in the first or only transmission chain of the message.

MFS uses the following ATTACH and SCHEDULER FM header parameters:

- ATTDP and ATTIU. These ATTACH parameters indicate whether the message is demand-paged or whether it is single or multiple chain.
- ATTDBA. This ATTACH parameter indicates the output data blocking algorithm or input data deblocking algorithm to be used by IMS. The data entity deblocked on input becomes the input data record to MFS. MFS then produces a standard IMS message consisting of pages and segments. On output, the MFS record becomes the data entity for output VLVB blocking. An exception is MFS DPM stream mode, which is sent as chain (ATTDBA) output.
- ATTDPN/SCDDPN and ATTPRN/SCDPRN. The ATTDPN/SCDDPN parameter is used on input to activate MFS input formatting. The ATTPRN/SCDPRN can be inserted into the input data stream and, therefore, made available to the application. On output message replies from IMS, the ATTDPN/SCDDPN or ATTPRN/SCDPRN can be inserted into the ATTACH or SCHEDULER FM headers of the output message processed by MFS as specified by MFS format descriptions. These override the ATTRDPN/SCDRDPN and ATTRPRN/SCDRPRN that might have been wrapped from the source input message ATTACH or SCHEDULER FM headers.
- ATTRDPN/SCDRDPN. The ATTRDPN/SCDRDPN can optionally be inserted into the input data stream and, therefore, made available to the application for input messages processed by MFS DPM. The return destination process name can be inserted into the output ATTACH or SCHEDULER FM header of the output message processed by MFS. The MFS-suggested return destination process to be associated with a reply to the output message is the MID name specified in the NXT= operand of the MOD.
- ATTRPRN/SCDRPRN. The return primary resource name (RPRN) can be inserted into the ATTACH or SCHEDULER FM header of the output message processed by MFS. It can be specified using MFS definition. If returned to IMS with a reply to the message on which it was sent, it determines the destination — transaction or LTERM — that is to receive the input reply.
- ATTDQN/SCDDQN. The destination queue name is sent by IMS on demand-paged output as a message identifier and should be returned on all requests associated with demand-paged messages.

When the first chain (ATTACH SCHEDULER) of an asynchronous demand-paged output message is successfully sent, the output queue (IMS LTERM) is automatically locked against input or output. This

function is also automatic following a QSTATUS FM header sent by IMS because of a cursor error on a demand-paged request. The queue lock state is reset when the page request (ATTACH QMODEL, QGETN with QNAME) is received and the first page of data is sent without resending the ATTACH or SCHEDULER FM header. The queue lock condition is also reset across session failure and subsequent restart. During the time that IMS waits for this page request (QGETN), provided conversation or response mode is not used, IMS can send or receive messages for other LTERMs. During the time that the output LTERM is locked, no input is accepted from the ISC session that produces output to the locked queue because of the input/output component relationships defined for the locked queue using the IMS system definition NAME macro statement or an ETO logon descriptor. Input is accepted that produces output for other LTERMs.

Data descriptor FM headers

The input and output data descriptor FM headers are used to receive and send a data structure name and version ID or a field tab separator character.

Input data descriptor FM header

On input to IMS, this header can follow the ATTACH or SCHEDULER FM header and is used to receive a data structure name (if DPAGE selection is by DPAGE name) and version ID or the input field tab separator character.

The input data descriptor FM header should be sent to IMS if input DPAGE selection is to be performed on the DPAGE name; that is, if you specify OPTIONS=DNM on the DIV statement (TYPE=INPUT).

If you set OPTIONS=DNM, and specify no DPAGE name or the wrong DPAGE name, an error occurs.

The version ID should be sent to IMS in the first or only FM header of the input message if MFS is to verify that the correct definition is used to map the data. If the version ID is sent and is X'0000', no verification occurs. The field tab separator is not required on input, because MFS provides for up to eight separators with the FTAB function. If received on input, the field tab separator is used instead of the MFS FTAB specification (if any) for the current transmission.

Related reference

“Data descriptor FM header formats” on page 565
The following tables show the formats of the input and output data descriptor FM headers.

Output data descriptor FM header

On output from IMS, the data descriptor FM header is used to send a data structure name and version ID or the output field tab separator character if OFTAB= parameter of the DIV or the DPAGE statement is specified.

It is sent in the only transmission chain of a nonpaged output message or in each transmission chain of a paged output message. For a nondemand-paged message (OPTIONS=MSG), this header follows the ATTACH or SCHEDULER FM header. For an autopaged message, this header follows the ATTACH or SCHEDULER FM header and precedes the data for the first logical or presentation page. The data descriptor FM header precedes the data for each additional page of output and is the only FM header in the transmission chain. For demand-paged output, this header follows the QFXR header. The version ID is sent only once for each message in the first or only FM header for the output message. Additionally, if OFTAB is specified and OPTIONS=DNM is requested, the output field tab separator character used for the current transmission is included in the FM header. If OPTIONS=NODNM is specified, a data descriptor FM header is not sent, regardless of whether OFTAB has been defined or not.

Related reference

“Data descriptor FM header formats” on page 565

The following tables show the formats of the input and output data descriptor FM headers.

Controlling demand-paged messages: QMODEL FM headers

Queue model (QMODEL) headers are sent and received to control demand-paged messages. All demand-paging requests (except RAP) must be made using the QMODEL-defined FM headers.

Restriction: MFS DPM-Bn does not support operator control table requests or input messages while sending demand-paged output.

The QMODEL headers that IMS supports are listed in the following table.

Table 103. IMS-supported QMODEL headers

QMODEL headers	Sent by IMS[“1” on page 552]	Received by IMS[“2” on page 552]
QXFR	X	
QGETN		X
QGET		X
QPURGE		X
QSTATUS	X	

Notes:

1. QMODEL headers are not sent between two IMS subsystems.
2. A QMODEL FM header must precede a QMODEL page request for the first page. These FM headers can be preceded by the QMODEL ATTACH FM header with the DPN parameter containing the value X'03'.

QXFR is a QMODEL reply. All other headers are QMODEL requests. Each half session between the logical units might indicate, by using the session bind parameters, whether the QMODEL process is available to receive QMODEL requests. The bind parameters have no effect on QMODEL replies. IMS always indicates "QMODEL available" and is prepared to receive QMODEL (paging) requests to IMS demand-paged output. The other half session might or might not indicate "QMODEL available." IMS ignores the other half-session bind indication for QMODEL requests, because only QMODEL replies can be sent by IMS.

Queue model uses LU 6.1 protocols to handle messages as follows:

- Although SNA allows multiple messages to be active, IMS messages are serial. That is, only one message is active at any given time. When active message processing is completed, another message can be processed. Synchronous (ATTACH without SCHEDULER) output MFS DPM demand-paged messages are considered active immediately when the first chain (ATTACH) is sent. Demand-paged output sent with ATTACH SCHEDULER does not become active until the first page request is returned to IMS.
- MFS demand-paged output provides for two types of message organization: linear (OPTIONS=DPAGE) and hierarchic (OPTIONS=PPAGE). For both types, pages can be retrieved sequentially (get next) or linearly (get by cursor). However, linear retrieval is permitted only if operator logical paging (OLP) is defined. Hierarchic retrieval is not supported.

Related reference

“Data flow control protocol reference” on page 515

The following topics describe the byte-level protocols for data flow control (DFC). The protocols are presented here.

Request (input) QMODEL FM headers

The following topics describe the QGETN, QGET, and QPURGE FM headers.

QGETN FM header

A QGETN FM header is received by IMS to cause sequential transfer of a single logical or presentation page of MFS DPM demand-paged output. It is the only QMODEL FM header in the chain and is never followed by function management data.

The QGETN FM header is not sent by IMS.

The QGETN FM header can be sent to IMS in the following cases:

- To retrieve first page of demand-paged output. A QGETN must always be the first request following the receipt of a scheduled demand-paged output. It must be preceded by an ATTACH QMODEL FM header.
- Following a QSTATUS FM header that indicates an invalid cursor in scheduled demand-paged output.
- When a non-OLP demand-paged output message is in progress.
- When an OLP demand-paged output message is in progress and the current cursor position is not at the last page of the message.

Each of these cases causes a QXFR FM header to be returned in reply.

The QGETN FM header cannot be sent to IMS in the following cases:

- Between messages. An error message is returned.
- While an OLP demand-paged output message is in progress and the current cursor position is at the last page.

Related reference

"QSTATUS FM header" on page 554
A QSTATUS FM header is sent by IMS in reply to an input QPURGE FM header or because an invalid cursor was detected on a page request for a demand-paged output.

"QGETN FM header format" on page 568
The following table shows the format of the QGETN FM header.

QGET FM header

Use the QGET FM header as input to IMS to transfer to the next, the last, or any logical page, according to cursor.

A QGET FM header is only valid for demand-paged output with operator logical paging defined. It is the only FM header in the input chain and is never followed by function management data.

The QGET FM header is not sent by IMS.

The QGET FM header cannot be sent to IMS in the following cases:

- Between messages. An error message is returned.
- After receiving the ATTACH for a scheduled demand-paged output message. The first paging request for this type of output must be QGETN. QGET can be used after requesting the first output page by QGETN.
- While a non-OLP demand-paged output message is in progress. An error message is returned. The output message is placed on the queue for retransmission.
- When the cursor value does not contain a valid 2-byte binary number. An error message is returned, and the output message is placed on the message queue for retransmission.
- When the cursor value is outside the range for the output message.

Related reference

"QSTATUS FM header" on page 554
A QSTATUS FM header is sent by IMS in reply to an input QPURGE FM header or because an invalid cursor was detected on a page request for a demand-paged output.

"QGET FM header format" on page 567
The following table shows the format of the QGET FM header.

QPURGE FM header

When IMS receives a QPURGE header, it halts processing on the demand-paged output message in progress.

QPURGE can be received by IMS only while an MFS demand-paged output message is active. The QPURGE FM header is the only FM header in the chain and is never followed by function management data. IMS replies to a QPURGE FM header with a QSTATUS. If IMS receives a QPURGE header to delete the demand-paged output message, the message is deleted only after IMS receives a sync-point response in reply to QSTATUS.

The QPURGE FM header is not sent by IMS.

Related reference

"QPURGE FM header format" on page 569
The following table shows the format of the QPURGE FM header.

Reply (output) QMODEL FM headers

The following topics describe the QXFR and QSTATUS FM headers.

QXFR FM header

The QXFR header is sent in each output transmission chain for a demand-paged output message in reply to a valid paging request by a QMODEL header.

A QXFR FM header can be followed by the data descriptor (DD) FM header (based upon MFS format definitions) and carries logical or presentation page data. This header must not be sent to IMS.

Related reference

"QXFR FM header format" on page 571
The following table shows the format of the QXFR FM header.

QSTATUS FM header

A QSTATUS FM header is sent by IMS in reply to an input QPURGE FM header or because an invalid cursor was detected on a page request for a demand-paged output.

If a QSTATUS results from a QPURGE during demand-paged output, it is sent to request a sync point. The message is dequeued when the sync-point response is received. The message is returned to the queue and made available for retransmission if an exception sync-point response is received.

A QSTATUS resulting from an invalid cursor during scheduled demand-paged output is sent RQE1/CD and allows a subsequent QGETN to again request the first page of the message or a QPURGE to cause the message to be dequeued.

The QSTATUS FM header is the only FM header in the chain and is not followed by function management data. The QSTATUS FM header must not be sent to IMS.

Related reference

"QSTATUS FM header format" on page 570

The following table shows the format of the QSTATUS FM header.

The RAP FM header under MFS

When used by MFS, the reset attached process (RAP) FM header can be sent to IMS to delete the demand-paged output message in progress (prevent further processing).

This header is equivalent to the MFS operator control function of NEXTMSG. This header can also be sent to IMS to delete the operator logically paged message in progress. A RAP FM header cannot be followed by function management data. The RAP FM header is not sent by IMS.

Related reference

“Reset attached process (RAP) FM header format” on page 573
The following table shows the format of the Reset Attach Process (RAP) FM header.

Chapter 32. FM header format reference

Each header's length is defined by a 1-byte length field. The value in this field includes the length field itself. A concatenation flag indicates whether an additional header follows.

Within the header formats, variable- and fixed-length parameters are positional by command code. A 1-byte length field precedes each variable-length positional parameter. The value contained in this length field can be X'00' through X'08' and does not include the length field itself. If the length field contains X'00', the variable parameter is omitted and the next positional variable-length parameter length field occurs followed by its variable-length parameter field.

Trailing positional parameter length fields of X'00's at the end of the header can be eliminated for input to IMS and are not sent on output from IMS. IMS also does not send trailing blanks for any of these names.

When receiving a header, IMS ignores any coded parameters beyond the last parameter to which it is sensitive (that is, that IMS supports). The other LU 6.1 subsystem should do the same when receiving headers that IMS sends.

Related reference

"ISC data flow control examples" on page 643
The following topics provide examples of ISC data flow control.

"Examples using ISC edit ATTACH parameters" on page 577
The following topics provide examples of using ISC edit ATTACH parameters.

ATTACH FM header format

The format of the ATTACH FM header is defined in the following table.

Table 104. Attach FM header format

Byte	Bits	Name	Content
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000101'
2-3		FMH5CMD	X'0202'

© Copyright IBM Corp. 1974, 2020

Table 104. Attach FM header format (continued)

Byte	Bits	Name	Content
4	0-3	FMH5MOD	
	4		Reserved
	5	ATTDP	Session Local Flag **B'0'** Not demand-paging **B'1'** Demand-paging
	6-7	ATTIU	Interchange Unit code **B'00'** Multiple chain **B'01'** Single chain **B'10'** Reserved **B'11'** Reserved
5		FMH5FXCT	Length of fixed-length parameters must be X'02'
6		ATTDSP	One-byte hexadecimal input value into a data stream profile. (IMS requires a value from X'00' to X'03' to select components 1-4 respectively. Value X'00', or component 1, is assumed if no ATTACH FM header is received while the Attach manager is in reset state.)
7		ATTDBA	Application data handling algorithm **X'00'** Undefined (Input only to IMS. IMS also assumes this as the default value if no ATTACH FM header is received while the attach manager is in reset state.) **X'01'** Variable length, variable blocked (input/output for IMS) **X'02'** Document subset of SCS (not supported by IMS) **X'03'** Card subset of SCS (not supported by IMS) **X'04'** A chain of RUs (input to/output from IMS) **X'05'** An RU (input only to IMS) **X'06' to X'FF'** Reserved

Table 104. Attach FM header format (continued)

Byte	Bits	Name	Content
8-m		ATTDPN[“1” on page 559]	Name of a process to be initiated. For an SNA-defined process, the DPN is a 1-byte, nongraphic hexadecimal character. Supported SNA processes are: **X'01'** SYSMSG **X'02'** SCHEDULER **X'03'** QMODEL The IMS processes are described under the ATTDPN parameter.
m+1-n		ATTPRN[“1” on page 559]	Name of primary resource for the process being initiated
n+1-p		ATTRDPN[“1” on page 559]	Name of suggested return process name
p+1-q		ATTRPRN[“1” on page 559]	Name of suggested primary resource for the return process
q+1-r		ATTDQN[“1” on page 559]	Name of queue to be associated with the attached process
r+1-s		ATTACC	Access code—Ignored by IMS.

Notes:

1. If the ATTDPN is set to SCHEDULER (X'02'), the remaining ATTACH parameters must not be specified. In this case, an equivalent for each of the asterisked parameters might occur in the concatenated SCHEDULER FM header. These SCHEDULER parameters are prefixed by "SCD" rather than by "ATT", but follow the same definition and rules as those for ATTACH.

ATTIU

The interchange unit code (ATTIU) indicates whether a single input or output IMS message consists of one or more SNA chain.

All input messages that are not MFS-autopaged messages might indicate either single or multiple chain, but are restricted to one actual transmission chain indicating end-of-message. MFS-autopaged input messages might indicate multiple chain regardless of whether the message is actually one or more than one SNA chain. MFS also offers an option that permits single chain messages to produce multiple DPAGEs. Non-MFS and MFS output that is not autopaged is always sent as a single transmission chain with ATTIU indicating single chain. MFS-autopaged output messages are always sent with ATTIU indicating multiple chain.

Restriction: IMS does not support any relationship between consecutive input or output messages. Therefore, the interchange unit code indicating multiple chain cannot be used to send IMS batched input messages or consecutive messages to be scheduled using the SCHEDULER process.

IMS requires attached (synchronous) input message switches to be one chain with at least EB specified.

Restriction: IMS does not support MFS-autopaged synchronous input message switches of more than one SNA chain.

The ATTIU parameter is not automatically retained from one input or output message to another. This parameter must be explicitly provided with each input message, or by MFS if the output message is multichain and is formatted by MFS. The input reset state for ATTIU is "multiple chain."

ATTDSP

The data stream profile (ATTDSP) parameter is ignored when attaching QMODEL (ATTDPN=X'03'). During other input, the ATTDSP field indicates a specific IMS input component.

The input component is used to identify an input IMS LTERM from a set of LTERMs (LTERM subpool) allocated to the session. The input DSP value must indicate a valid input component (an IMS LTERM defined with the input component value), or the input is rejected.

Several internal IMS functions are associated with the LTERM defined for the input component. For example, the LTERM defines a particular origin (input) and destination (output) path within IMS. This output destination can be the same as, or different from, the input origin LTERM and represents a stable application and operator reference point within IMS.

For statically defined terminals, input terminal security can be defined for the input LTERM. This security authorization can be unique for each LTERM or can represent a security level, or group, within IMS. Input terminal security authorizes access to IMS terminals, transactions, and commands.

The input ATTACH DSP parameter remains in effect until it is:

- Changed by another ATTACH
- Reset by a system failure
- Backed out to its value at the last sync point by the ERP
- Reset at between-brackets state
- Reset by the RAP FM header

The input reset state for the ATTDSP parameter is X'00' or "component 1."

Because IMS does not remember the active input ATTDSP across IMS system failures, the ATTDSP must be provided using an ATTACH FM header to restore the receiving process after a bind or negotiable bind response indicating "in-brackets" (possible restart).

The output LTERM is determined by either the LTERM selected for input or by the message sender, as for message switches and broadcast messages. IMS also allows the receiving message processing program (MPP) to change the destination of resulting output by issuing a CHNG call and inserting the output to a modifiable alternate PCB. For all system messages resulting directly from input (for example, commands), the output LTERM selected is the normal destination LTERM defined for the LTERM from which the input originated.

During output, the DSP field is set to the output component value associated with the output LTERM.

IMS Message Format Service can be defined as available on a component-by-component basis. The output component also specifies the bracket and send/receive protocol to be used for asynchronous output sent using the ATTACH having a DPN parameter indicating "SCHEDULER."

Related concepts

“Coordinating the restart process” on page 495
Half sessions use rules to coordinate restart after a session failure.

ATTDBA

The deblocking algorithm (ATTDBA) determines the data blocking and deblocking algorithm to be used.

This algorithm, along with an indication of whether MFS is being used, determines the amount of data presented to or from a process for a single get or put operation.

When IMS is sending messages and MFS is not being used, each segment of the message on the message queue becomes a data entity within the RU (or chain of RUs). If a segment of the message spans message queue records (LRECLs), then each spanned portion of the segment becomes a data entity. If MFS is used, the segments are blocked and sent as a data entity according to the definition of the MFS record.

When IMS receives messages, each data entity of the input message is inserted in the message queue as a segment of the message. Because ISC does not support spanned queue segments on input, the

largest message queue LRECL must be large enough to handle the largest data entity or MFS data record received. If this LRECL definition is exceeded, an error is detected and an appropriate error message is produced (DFS074). Also, the actual space available in the message queue is reduced by any variable prefix items on the message.

IMS implements four of the algorithms defined by SNA:

UNDEFINED
Same as RU, below, for IMS.

RU
Input RU is the entity presented to the attached process. IMS receives this DBA value but does not send it.

VARIABLE LENGTH, VARIABLE BLOCKED (VLVB)
Each data entity sent or received is preceded by a 2-byte length field that allows it independence from RU size or boundaries. Several data entities can be blocked into a single RU, or a data entity can span RUs. The length includes the 2 bytes for the length field itself.

CHAIN OF RUs
Each data entity is sent or received as a single SNA chain.

Each message received by IMS is deblocked based on the ATTDBA field in the ATTACH FM header. The input ATTDBA field value must indicate a supported algorithm, or the input is rejected.

The input ATTACH DBA parameter remains in effect until it is:

- Changed by another ATTACH
- Backed out to its value at the last sync point by the ERP
- Reset at between-brackets state
- Reset by the RAP FM header
- Reset by a session failure

The input reset state for the ATTDBA parameter is "UNDEFINED" (equivalent to "RU").

Because IMS does not remember the active input ATTDBA across IMS system failures, the ATTDBA must be provided using an ATTACH FM header to restore the receiving process after a bind or negotiable bind response indicating "in-brackets" (possible restart).

With the exception of output using MFS stream mode, output from IMS is automatically sent indicating the variable-length, variable blocked (VLVB) algorithm in the ATTACH or SCHEDULER FM header. This output is sent indicating chain assembly.

Related concepts
“Coordinating the restart process” on page 495
Half sessions use rules to coordinate restart after a session failure.

ATTDPN

The destination process name (ATTDPN) parameter identifies, explicitly or implicitly, the input process to be attached to the half session.

One responsibility of the attached process is to determine, for the receiving subsystem, the destination of the input message within that subsystem. In some cases, the named process might be the message destination. Within IMS, the receiving process (except basic edit) uses the input primary resource name (ATTPRN), if provided, as the message destination transaction queue or LTERM. If an ATTPRN is not provided, the standard IMS algorithm for determining message destination is used.

Related reading: For more information on the IMS algorithm for determining message destination, see *IMS Version 15 System Administration* under "IMS Messages and Their Scheduling."

All attached processes execute synchronously with the session. However, some processes can schedule additional work to be done asynchronously within IMS. The process attached remains attached until it is:

- Changed by another ATTACH

- Backed out to its value at the last sync point by the ERP
- Reset by the end-bracket indicator
- Reset by the RAP FM header
- Reset by a system failure

The input reset state for the ATTDPN parameter is "ISCEDT".

Because IMS does not remember the active input ATTDPN across IMS system failures, the ATTDPN must be provided using an ATTACH FM header to restore the receiving process after a bind or negotiable bind response indicating "in-brackets" (possible restart).

The ATTDPN parameter might indicate that the process to be attached is one of the following:

ISC edit (‘ISCEDT' or user-named alias)
IMS Basic Edit (‘BASICEDT')
MFS formatting (MFS MID name)
System Message (SYSMSG), X'01'
SCHEDULER model, X'02'
Queue model (QMODEL), X'03'

With the exception of MFS and QMODEL, these processes are always available. MFS is only available if defined for the input component (ATTDSP) during IMS system definition. If the ATTDPN is an MFS MID name and MFS is not available, the input message is rejected. QMODEL is available only following an MFS demand-paged output message.

ISCEDT (as well as the alias defined by the user for ISCEDT during IMS system definition) and BASICEDT are reserved names, and cannot be used as MFS MID or MOD names. Use of these names results in the use of the named process rather than an MFS process.

ISC edit is selected if no ATTACH FM header (or no ATTDPN) is supplied when the Attach manager is in reset state. The "active" process is used if no ATTACH FM header (or no ATTDPN) is supplied when the attach manager is not in the reset state.

IMS sets the output ATTDPN on a return reply to the value contained in the ATTRDPN parameter optionally provided within the ATTACH of the previous input request. When sending system messages, IMS automatically inserts the SYSMSG ATTDPN where necessary. MFS DPM provides a way to optionally specify, override, or delete the output ATTDPN by using the output message format descriptor. When the ATTDPN parameter is not available for output, a 1-byte field containing X'00' is included within the output ATTACH FM header.

Related reference

“ATTPRN” on page 562
The ATTACH primary resource name (ATTPRN) parameter represents the destination for an input message in the receiving subsystem.

“ATTRDPN and ATTRPRN” on page 563
The return destination process name (ATTRDPN) and return primary resource name (ATTRPRN) parameters define the reply process and the return primary resource within the source session and should be returned to the source session on resulting replies to facilitate return-reply routing within the source session.

ATTPRN

The ATTACH primary resource name (ATTPRN) parameter represents the destination for an input message in the receiving subsystem.

This parameter is sent on a return reply from a remotely executed message that results from the returning reply message of the ATTACH ATTRPRN parameter that was sent on a message request. However, in situations where the message request contains no input ATTRPRN, the output ATTPRN parameter can be created using MFS, based on user-defined information.

IMS does not permit specification of IMS command verbs as PRNs.

This destination normally represents a terminal (an LTERM), rather than an application program. If the request ATTRPRN/reply ATTPRN represents an application program, consider the synchronous versus asynchronous relationship between the source application program and the remote transaction executions.

Further, you should consider input transaction security requirements for processing and routing the reply. If supplied on input to IMS, the ATTPRN is used as an IMS transaction code or as the LTERM name for a message switch. The data stream is not edited for destination and security information. If the ATTPRN represents a transaction, and if password security (which applies to statically defined terminals only) was defined for the node, an input security error results, because no available source exists for the input password.

Related reading: For more information on password security, see *IMS Version 15 System Administration*.

The ATTPRN is rejected during IMS conversation mode when an application has previously inserted the transaction code into the SPA. The ATTPRN only applies to a single message instance and overrides, but does not destroy, any "preset" destination established using an IMS **/SET** command. Subsequent messages with no ATTACH FM header or with an ATTACH FM header where the ATTPRN parameter is not supplied, can use any preset destination previously established by the IMS **/SET** command or by an IMS transaction code, LTERM name, or command verb supplied in the data or through MFS formatting.

IMS sets the output ATTPRN on a return reply to the value contained in the ATTRPRN parameter optionally provided within the ATTACH of the previous input request.

MFS DPM provides a means to optionally specify, override, or delete the output ATTPRN using an output message format descriptor. When this parameter is not present, a 1-byte field containing X'00' is sent with the output ATTACH FM header.

The ATTPRN is not automatically retained from one input or output message to another. This parameter must be explicitly provided with each input message, or through the output MFS descriptors if the output message is processed by MFS.

Related reference

“ATTRDPN and ATTRPRN” on page 563
The return destination process name (ATTRDPN) and return primary resource name (ATTRPRN) parameters define the reply process and the return primary resource within the source session and should be returned to the source session on resulting replies to facilitate return-reply routing within the source session.

ATTRDPN and ATTRPRN

The return destination process name (ATTRDPN) and return primary resource name (ATTRPRN) parameters define the reply process and the return primary resource within the source session and should be returned to the source session on resulting replies to facilitate return-reply routing within the source session.

If provided on input to IMS and not changed or deleted by an output MFS format description, these parameters are returned to the source session unmodified in the reply ATTACH FM header as the output ATTACH ATTDPN and ATTPRN parameters respectively. These parameters are associated with the message to be processed and are recovered with the message across session and subsystem failures.

The input ATTRDPN and ATTRPRN are not automatically retained from one input or output message to another. The ATTRDPN and ATTRPRN must be explicitly provided with each input message or through the MFS output format descriptors when MFS DPM is used for output. The ATTRDPN results from MFS if a next MID was specified in the MOD.

A different procedure is followed when the reply is to be returned on a session different from the session on which the input message originated. If the output ATTRPRN is not set by MFS, and the source of the output message is not associated with the same session, IMS does not wrap the input ATTRDPN and ATTRPRN as the output ATTDPN and ATTPRN respectively. In this case, IMS also automatically inserts the source LTERM name of the terminal entering the input message switch or transaction (for alternate PCB output) as the output ATTRPRN parameter.

When the ATTRDPN parameter is not available for output, a 1-byte length field containing X'00' is included in the output ATTACH FM header.

The following tables summarize IMS actions relative to the DPN, PRN, RDPN, and RPRN fields in the ATTACH and SCHEDULER headers sent with the message.

Table 105. IMS interpretations for the DPN, PRN, RDPN, and RPRN fields

Input FMH	IMS interpretation
DPN	MFS MID name or input message editor name (ISC edit or basic edit). Defaults to ISCEDT if not supplied.
PRN	Input transaction code or LTERM name, overriding data stream and preset mode except during conversation.
RDPN	Saved as default reply DPN field.
RPRN	Saved as default reply PRN field.

Table 106. IMS actions for the DPN, PRN, RDPN, and RPRN fields

Output FMH field	IMS ACTION reply message sent on same session as input	IMS ACTION reply message or asynchronous output sent on another session
ATTDPN	Wrapped input ATTRDPN if provided; optionally overridden by MFS. For ATTACH-only output, if no parameter is available, the DPN is not sent. For SCHEDULER output, the default is ISCEDT.	Provided only using MFS. For ATTACH-only output, if no parameter is available, the DPN is not sent. For SCHEDULER output, the default is ISCEDT.
ATTPRN	Wrapped input ATTRPRN; optionally overridden by MFS	Provided only using MFS
ATTRDPN	Provided only using MFS	Provided only using MFS
ATTRPRN	Provided only using MFS	Automatically defaulted to source LTERM name; optionally overridden by MFS

ATTDQN and ATTDP

The destination queue name (ATTDQN) parameter names a specific message instance. This parameter is valid only for output MFS demand-paged messages.

For MFS demand-paged output, ATTDQN is sent by IMS as an output message identifier. This name can then be used by the other half session as the QNAME parameter within paging requests to access the IMS demand-paged output message. Within IMS, only one message can be active for a given session at any one time. This means that after output paging begins on a message and until the paging operation is terminated, all input must be paging requests indicating the same QNAME (from ATTDQN) value. The session-local flag (ATTDP) in the ATTACH FM header is set to 1. ATTDQN and ATTDP are not sent or received under other conditions.

ATTACC

IMS does not send the access code (ATTACC) parameter during output and ignores this parameter on input.

Data descriptor FM header formats

The following tables show the formats of the input and output data descriptor FM headers.

Table 107. Input data descriptor FM header format

Byte	Bits	Name	Contents
0	0-7	FMHL	Length
1	0	FMHC	
	1-7	FMHT	B'0000100'
2	0-7	FMH4FXCT	Fixed-length parm (X'03')
3	0-7	FMH4DTYP	
		FMH4UNDF	X'00' Reserved
		FMH4FIX	X'40' Field formatted record (FFR) - Fixed
		FMH4FXSP	X'41' FFR - All fields terminated by Separator (Note 1)
		FMH4MXSP	X'42' FFR - Fields terminated by Separator or Length defined by map X'43' - X'FF' Reserved (Note 1)
4		FMH4SEP	Separator character (Note 2)
5	0-7	FMH4PCTL	Presentation control byte (Note 3)
6-m		FMH4DSN	Data structure name (Note 4)
m+1-n		FMH4BDT	Block data type (Note 3)
n+1-p		FMH4VERS	Version ID (Note 5)

Notes:

1. From DIV or DPAGE statement.
2. Not required, because it is received on MFS format description.
3. Ignored by IMS if sent.
4. DPAGE name if DPAGE selection is to be performed on the data structure name. FMT name if OPTIONS=MSG. Required field.
5. Version ID to be used by MFS to verify that the correct definition is used to map the data. This field should be present only once for each input message if multiple transmission chains are used to create the input message. If present once for each message, it should be sent in the first transmission chain of the message. If the version ID is not sent or it is X'0000', MFS descriptor verification is not performed.

Table 108. Output data descriptor FM header format

Byte	Bits	Name	Contents
0	0-7	FMHL	Length
1	0	FMHC	
	1-7	FMHT	B'0000100'

Table 108. Output data descriptor FM header format (continued)

Byte	Bits	Name	Contents
2	0-7	FMH4FXCT	Fixed-length parm (X'03')
3	0-7	FMH4DTYP	
		FMH4UNDF	X'00' Reserved
		FMH4FIX	X'40' Field formatted record (FFR) - Fixed
		FMH4FXSP	X'41' FFR - All fields terminated by separator
		FMH4MXSP	X'42' FFR - Fields terminated by separator or length defined by map X'43' - X'FF' Reserved
4		FMH4SEP	Separator character
5	0-7	FMH4PCTL	Presentation control byte (Note 1)
6-m		FMH4DSN	Data structure name (Note 2)
m+1-n		FMH4BDT	Block data type (Note 3)
n+1-p		FMH4VERS	Version ID (Note 4)

Notes:

1. Set to 0 on output.
2. FMH4DSN is:
   - The FMT name if OPTIONS=MSG
   - The DPAGE name if OPTIONS=DPAGE
   - The PPAGE name if OPTIONS=PPAGE
3. Omitted if version ID not present. Set to 0 if version ID present.
4. Version ID specified in FMT definition or calculated by MFS if the default is used. This field is present only in the first or only data descriptor FM header for the message.

Error recovery procedure (ERP) FM header

The following table shows the format of the ERP FM header.

Table 109. Error recovery procedure (ERP) FM header

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHTYPE	X'07'
2-5		ERPSENSE	SNA sense code that would appear on error response
6-7		ERPSEQ	Sequence number of chain for which error was detected

QMODEL FM headers

The beginning of each header is fixed-length. This fixed-length area is followed by a variable number of fixed-length subfields, followed by a variable number of variable-length subfields.

QGET FM header format

The following table shows the format of the QGET FM header.

Table 110. QGET FM header format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A10'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-6		Reserved
	7	QGETLAST	B'1' Coded request, get last"1" on page 567
5		FMH6FXCT	Length of Fixed-Length Parameters (=X'01')
6		QORG	Type of paging request **X'00'** Queue organization not specified. Valid on input to IMS. **X'01'** Sequential. Invalid. **X'02'** Linear. Valid on input to IMS if it matches the QORG in the QXFER sent by IMS."2" on page 568 **X'03'** Hierarchic. Not supported by IMS. **X'04' to X'FF'** Reserved
7-m		QNAME	Q from which records are retrieved. IMS message ID (ATTDQN/SCDDQN)"3" on page 568
m+1-n		QCURSOR	Cursor vector in target queue"4" on page 568
n+1-p		QTRNSZ	Size of maximum record to be returned (binary). Ignored by IMS if sent. All fixed-length parameters must be present when you use this header.

Notes:

1. The QGETLAST selects a logical page whose value is equal to the last logical page in the message.

   The QGET FM header is valid and causes a QXFR reply only if the OLP demand-paged output message is in progress and the cursor value requested is within the range of the output message.

2. If QORG is invalid, an error message is sent and the output message is returned to the message queue for retransmission.
3. The QNAME (message identifier) is required on each page for scheduled demand-paged output and is optional for synchronous demand-paged output. If the QNAME parameter is present, it must match the ATTDQN/SCDDQN parameter in the ATTACH FM header sent by IMS. If QNAME is not specified for synchronous demand-paged output, IMS assumes it to be the same as the ATTDQN parameter name. If the QNAME parameter does not match the ATTDQN/SCDDQN parameter, an error message is sent and the output message is returned to the message queue for retransmission.
4. The cursor vector consists of one 2-byte binary number representing the logical page number. IMS does not support hierarchic retrieval; therefore, the 2-level cursor (logical page number followed by the presentation page number) is invalid and results in an error message being issued. The length value of the QCURSOR is 2. The next 2 bytes contain the logical page number (absolute) request.

QGETN FM header format

The following table shows the format of the QGETN FM header.

Table 111. QGETN FM header format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A10'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-7		Reserved
5		FMH6FXCT	Length of Fixed-Length Parameters (=X'01')
6		QORG	Type of paging request X'00' Queue organization not specified. This form is valid for both OLP and non-OLP output. **X'01'** Sequential."1" on page 569 **X'02'** Near."1" on page 569 **X'03'** Hierarchic. Not supported by IMS. **X'04' to X'FF'** Reserved.
7-m1		QNAME	Q from which pages are retrieved. Specifies a message ID (ATTDQN/SCDDQN)"2" on page 569
m+1-n		QTRNSZ	Size of maximum record to be returned (binary). Ignored by IMS if sent.

Notes:

1. If QORG is specified, the QORG specified on the resulting QXFR FM header sent by IMS must match it. Otherwise, an error message is sent and the output message is returned to the message queue for retransmission.
2. The QNAME (message identifier) is required on each page request for scheduled demand-paged output and is optional for synchronous demand-paged output. If the QNAME parameter is present, it must match the ATTDQN/SCDDQN parameter in the ATTACH FM header sent by IMS. If QNAME is not specified for synchronous demand-paged output, IMS assumes it to be the same as the ATTDQN parameter name. If the QNAME parameter does not match the ATTDQN/SCDDQN parameter, an error message is sent and the output message is returned to the message queue for retransmission.

QPURGE FM header format

The following table shows the format of the QPURGE FM header.

Table 112. QPURGE FM header format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A06'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-7		Reserved
5		FMH6FXCT	Length of fixed-length parameters (=X'01')
6		QORG	Type of queue purge function (if multiple DPAGE input) or paging request (if demand-paged output)"1" on page 569
7-m		QNAME	Name of IMS message ID (ATTDQN/SCDDQN)"2" on page 569

Notes:

1. If IMS receives QPURGE during demand-paged output, the QORG, if specified (that is, QORG is not equal to 0), must match the QORG specified in the QFXR FM header sent by IMS.
2. The QNAME (message identifier) is required on each page request for scheduled demand-paged output and is optional for synchronous demand-paged output. If IMS receives QPURGE during demand-paged output, the QNAME, if specified, must match the ATTDQN/SCDDQN parameter (message ID) in the ATTACH FM header sent by IMS. If QNAME is not specified for synchronous demand-paged output, IMS assumes the same name as the ATTDQN parameter name. If the QNAME parameter does not match the ATTDQN/SCDDQN parameter, an error message is sent and the output message is returned to the message queue for retransmission.

QSTATUS FM header format

The following table shows the format of the QSTATUS FM header.

Table 113. QSTATUS FM header format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command code: X'0A0A'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte length fields
	1-7		Reserved
5		FMH6XCT	Length of fixed-length parameters (=X'02')
6		QORG	Type of queue add/retrieval requests."1" on page 570
7	0-4		Not supported by IMS. Bit values should be B'0'.
	5	QINVCUR	B'0' if message is a reply to QPURGE. B'1' Invalid Cursor Indicates invalid logical page request received for OLP output.
	6-7		Not supported by IMS.
8			Reserved
9-12		QCURSOR	Current cursor value is 0"2" on page 570
13-16		QSENSE	Sense data"3" on page 570
17-n		QNAME	Message ID (ATTDQN/SCDDQN)

Notes:

1. If QSTATUS is sent by IMS in reply to the QPURGE, QORG is not specified (that is, QORG contains the value of zero). If QSTATUS is sent by IMS in reply to QGET by cursor request, QORG is set to linear (that is, QORG contains the value of two).
2. This parameter is present only if the QINVCUR value is set to B'1'. The cursor value is set to 0. A QGETN request retrieves the first page of the message for which this error (invalid cursor) was detected.
3. This parameter is present only if the QINVCUR value is set to B'1'. It consists of the DFS223 error message as a binary number.

QXFR FM header format

The following table shows the format of the QXFR FM header.

Table 114. QXFR FM header format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A08'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-7		Reserved
5		FMH6FXCT	Length of fixed-length fields (=X'02')
6		QORG	Types of paging requests valid for this message. **X'01'** Sequential retrieval of pages. Operator logical paging (OLP) is not defined. Message is dequeued after successful transmission of last page. **X'02'** Linear retrieval of pages. Operator logical paging (OLP) or browsing is allowed. Message is dequeued only by explicit action from the other subsystem (RAP or QPURGE FM header is received). **X'03'** Hierarchic retrieval of pages. Not supported by IMS. **X'00', X'04'-X'FF'** Reserved.
7	0-4	Reserved	
	5	QDISP	B'0' Disposition is save.
	6		Reserved.
	7	QEMSG	B'0' Not end of message B'1' End of message Set to B'1' if last logical page (OPTIONS=DPAGE) or last presentation page (OPTIONS=PPAGE) of a message is transmitted.

Table 114. QXFR FM header format (continued)

Byte	Bits	Name	Contents
8-n		QCURSOR"1" on page 572	Cursor vector for current message. For OPTIONS=PPAGE, this field includes a 1-byte length field of value 4 followed by the 2-byte current logical page number and the 2-byte current presentation page number. For OPTIONS=DPAGE, this field includes a length field of value 2, followed by the current logical page number."2" on page 572
n+1-n		QCOUNT"1" on page 572	Number of occurrences of pages at lowest level of cursor. If present, this field includes a length byte of value 2, followed by the defined number of pages in the current logical page. The QCOUNT field is present only if the defined number of presentation pages in the current logical page is greater than 1.
n+1-p		QRECLNG	Length of record before truncation. If 0, then either the record was not truncated or QTRNSZ on QGET(N) was ignored. This field is not supported by IMS and will not be included in the FM header.

Notes:

1. If operator logical paging (OLP) is allowed, the QCURSOR and QCOUNT fields contain information the receiver can use for formulating subsequent QGET by cursor requests.
2. Assume the following output message with OPTIONS=PPAGE:

   Logical Page 1 3 Presentation Pages Defined
   Logical Page 2 1 Presentation Page Defined
   Logical Page 3 2 Presentation Pages Defined

The following table illustrates the values placed in the QCURSOR and QCOUNT fields with sequential retrieval requests.

Table 115. QCURSOR and QCOUNT values with sequential retrieval requests

Transmission	Count/LP	No./PP No.	Qcursor Count/PP No.	Qcount comment
1	4 / 1	1	2 / 3	Logical Page 1
2	4 / 1	2	2 / 3	Logical Page 1
3	4 / 1	3	2 / 3	Logical Page 1
4	4 / 2	1	(omitted)	Logical Page 2
5	4 / 3	1	2 / 2	Logical Page 3
6	4 / 3	2	2 / 2	Logical Page 3

Note:

Output Message with OPTIONS=DPAGE:

Logical Page 1
Logical Page 2
Logical Page 3

The following table illustrates the values placed in the QCURSOR field with sequential retrieval requests.

Table 116. QCURSOR values with sequential retrieval requests

Transmission	Qcursor Count/LP	Qcount	Comment
1	2 / 1	Not Present	Logical Page 1
2	2 / 2	Not Present	Logical Page 2
3	2 / 3	Not Present	Logical Page 3

Reset attached process (RAP) FM header format

The following table shows the format of the Reset Attach Process (RAP) FM header.

Table 117. Reset attached process (RAP) FM header format

Byte	Bits	Name	Contents
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000101'
2-3		FMH5CMD	X'0204'
4		FMH5MOD	
	0	FMH5LNSZ	B'0': 1-byte parameter length field
	1-7		Length of parameter length fields B'0': 1-byte parameter length field B'1': Reserved
5		FMH5FXCT	Length of fixed-length parameters (X'00')

SCHEDULER FM header format

The following table shows the format of the SCHEDULER FM header.

Table 118. SCHEDULER FM header format

Byte	Bits	Name	Contents
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000110'
2-3		FMH6CMD	X'0802'

Table 118. SCHEDULER FM header format (continued)

Byte	Bits	Name	Contents
4		FMH6MOD	
	0	FMH6LNSZ	B'0': 1-byte parameter length field
	1	FMH6RPLY	**B'0'** No reply **B'1'** Reply[“1” on page 574]
	2	FMH6PROT	B'0' No protection;[“2” on page 575] B'1' Protection
	3	FMH6DELY	**B'0'** Timer optional **B'1'** Timer required[“1” on page 574]
	4-7		Reserved
5		FMH6FXCT	Length of fixed-length parameters - X'01'
6			Schedule Request control
	0	SCDTIME	Initiation delay specification **B'0'** Interval **B'1'** Time Not supported by IMS
	1-7		Reserved
7-m		SCDDPN[“3” on page 575]	Name of process to be initiated. (Required field)
m+1-n		SCDPRN[“3” on page 575]	Name of the primary resource for the process to be initiated
n+1-p		SCDRDPN[“3” on page 575]	Suggested name of the return process
p+1-q		SCDRPRN[“3” on page 575]	Suggested name of the primary resource for return process name
q+1-r		SCDDQN[“3” on page 575]	Name of queue associated with process
r+1-s		SCDREQN	Name of this process request instance (Not supported by IMS)
s+1-t		SCDDELY	Time interval to be decremented prior to initiation of destination process. (Not supported by IMS)

Notes:

1. Must be 0 on both input and output.

2. On input, this field is ignored. On output, this field is set to 0 for nonrecoverable output and to 1 for recoverable output.
3. See “ATTDPN” on page 561, “ATTPRN” on page 562, “ATTRDPN and ATTRPRN” on page 563, and “ATTDQN and ATTDP” on page 564 for a more detailed description of these parameters. In the ATTACH topic, these parameters are prefixed by "ATT" rather than "SCD". The rules for operation of these SCHEDULER parameters are the same as the rules defined for the ATTACH parameters.

SYSMSG process headers

The following tables show the formats of the SYSMSG process headers: the SYSERROR FM header and the SYSSTAT FM header.

SYSERROR FM header format

The following table shows the format of the SYSERROR FM header.

Table 119. SYSERROR FM header format

Byte	Bits	Name	Contents
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000110'
2-3		FMH6CMD	X'0404'
4		FMH6MOD	
	0	FMH6LNSZ	B'0': 1-byte parameter length field
	1-7		Reserved
5		FMH6FXCT	Length of fixed-length parameters (X'00')
6-m		ATTDPN	RDPN field supplied with an input message resulting in this SYSMSG
m+1-n		ATTPRN	RPRN field supplied with an input message resulting in this SYSMSG

SYSSTAT FM header

The following table shows the format of the SYSSTAT FM header.

Table 120. SYSSTAT FM header

Byte	Bits	Name	Content
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	B'0000110'
2-3		FMH6CMD	X'0402'
4		FMH6MOD	
	0	FMH6LNSZ	B'0' 1-Byte length fields
	1-7	Reserved	

Table 120. SYSSTAT FM header (continued)

Byte	Bits	Name	Content
5		FMH6FXCT	Length of fixed-length parameters

Chapter 33. Examples using ISC edit ATTACH parameters

The following topics provide examples of using ISC edit ATTACH parameters.

ATTACH and SCHEDULER parameters with ISC edit

The following series of figures illustrate the use of the ATTACH parameters when using ISC edit for IMS input and output messages.

ISC edit is used for normal transactions, commands, and message switches between LTERMs when one of the following occurs:

- An ATTACH FM header is received while in a between-brackets state that does not specify a destination process name (DPN).
- The DPN names IMS ISC edit as defined during IMS system definition.
- No ATTACH FM header is received either when the attach manager is in the between-brackets reset state or when the active process is ISC edit.
- The DPN=ISCEDT.

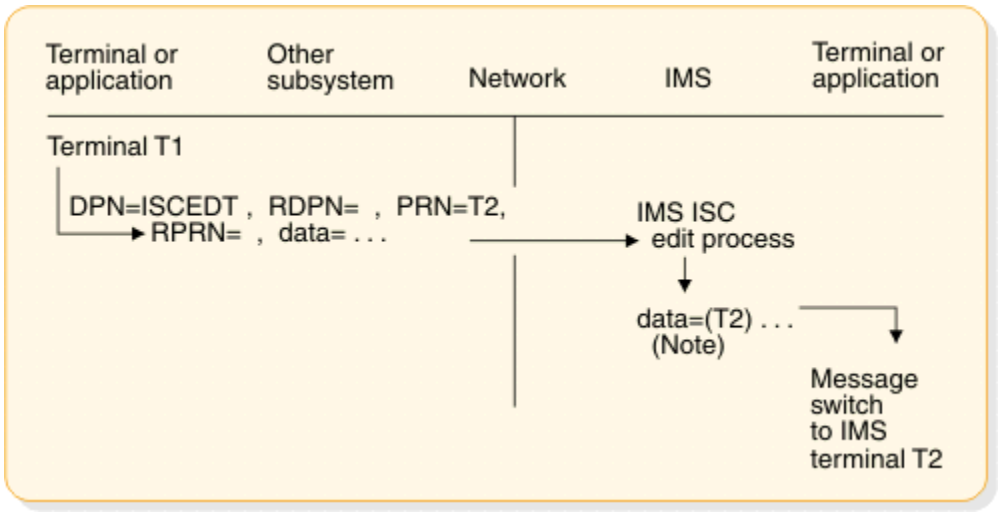


Figure 72. Example of message switch from another subsystem terminal to IMS terminal

Note: The IMS LTERM name for terminal T2 can be supplied either as the input primary resource name (PRN) or as the first field in the data.

© Copyright IBM Corp. 1974, 2020

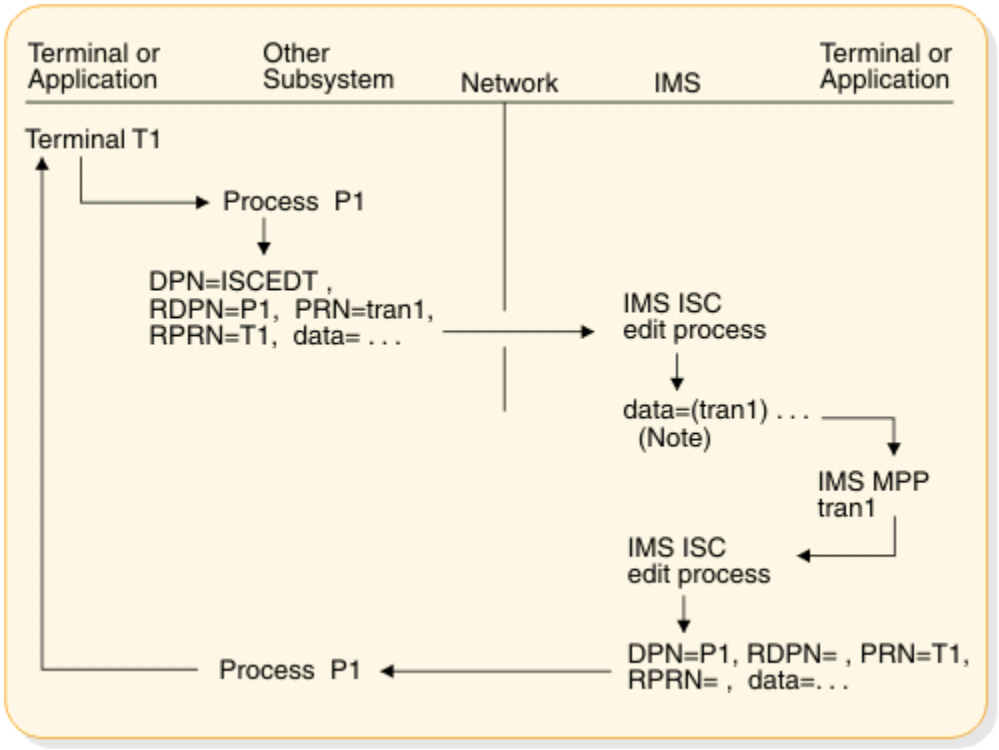


Figure 73. Example of another subsystem accessing an IMS application program with a reply back to the subsystem terminal

Note: The IMS transaction code can be supplied either as the input PRN or within the data. The input RDPN and RPRN parameters become the DPN and PRN parameters respectively for any resulting reply from the application program when not processed by MFS.

In the following figure, messages are routed from IMS Terminal 1 to the ISC edit process through either a message switch or an MPP or Fast Path message routing application. The ISC edit process uses SCHEDULER parameters to send the message to a transaction on another subsystem. The other subsystem then sends the message back to IMS and the ISC edit process using similar SCHEDULER parameters. The ISC edit process directs the message back to Terminal 1.

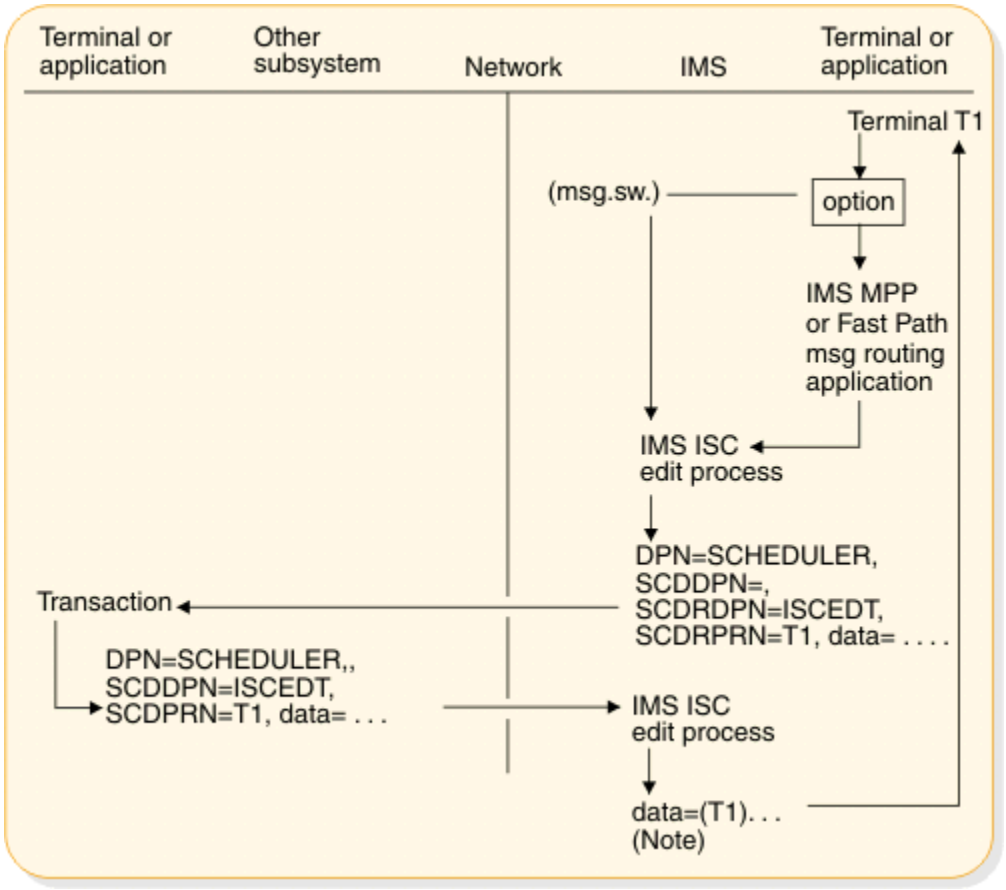


Figure 74. Example of message routing with ISC edit

Note: The IMS LTERM name can be supplied either as the input SCDPRN or the first data field.

ATTACH parameters with the IMS SYSMSG process

The system message process (SYSMSG) is indicated through a special SNA process name in the ATTACH header.

Input system messages for IMS ISC sessions are logged and routed directly to the IMS master terminal operator when no PRN is supplied using the input ATTACH parameters. If a PRN value is supplied for IMS ISC sessions, it becomes the IMS input message destination transaction code or LTERM name (message switch), and the input system message is passed to ISC edit. SYSMSG does not provide access to the IMS command processor.

A reply can result from the input SYSMSG if a PRN value supplied is for an IMS transaction. This reply is sent just as is any other (non-SYSMSG) asynchronous output message.

IMS requests attachment of the SYSMSG process to send IMS broadcast and other system messages that are not directly solicited by an input IMS command.

Exception: When IMS detects an error condition during an input IMS response mode or conversational transaction, an exception response (selective receiver ERP) to the input transaction occurs and the system message is sent as an ERP message.

The following figures illustrate the use of the ATTACH parameters when attaching the SYSMSG process for IMS input and output messages.

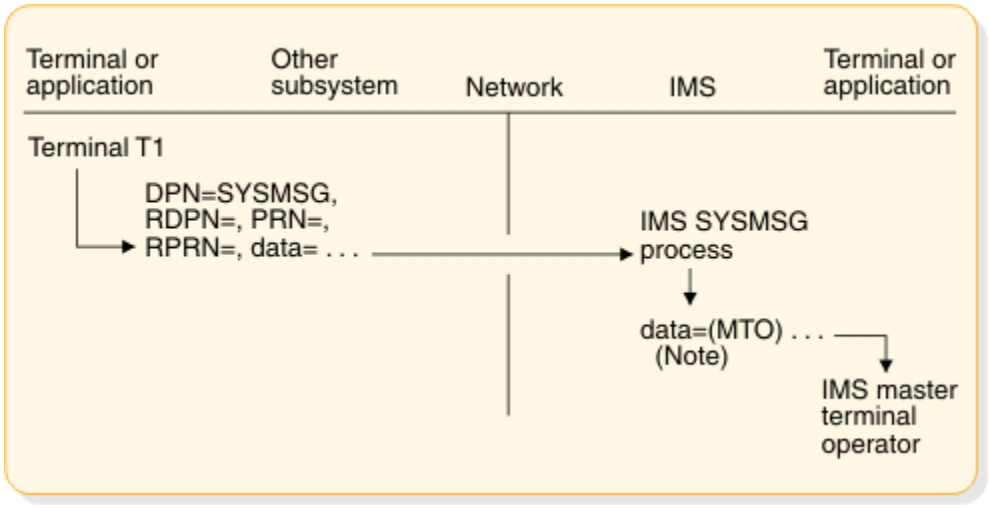


Figure 75. Example of SYSMSG without PRN from other subsystem terminal to IMS A

Note: The IMS LTERM name for the master terminal operator is assumed when no input ATTACH PRN parameter is supplied.

In the following figure, the input message from Terminal 1 is routed from a non-IMS subsystem to the IMS SYSMSG process in IMS using ATTACH parameters. The IMS SYSMSG process passes the message to the IMS edit process, which sends the data to tran1 in the IMS MPP region. After tran1 processes, the reply is passed to the IMS edit process again, which uses SCHEDULER parameters to send the reply back to Process 1 and then Terminal 1 on the other subsystem.

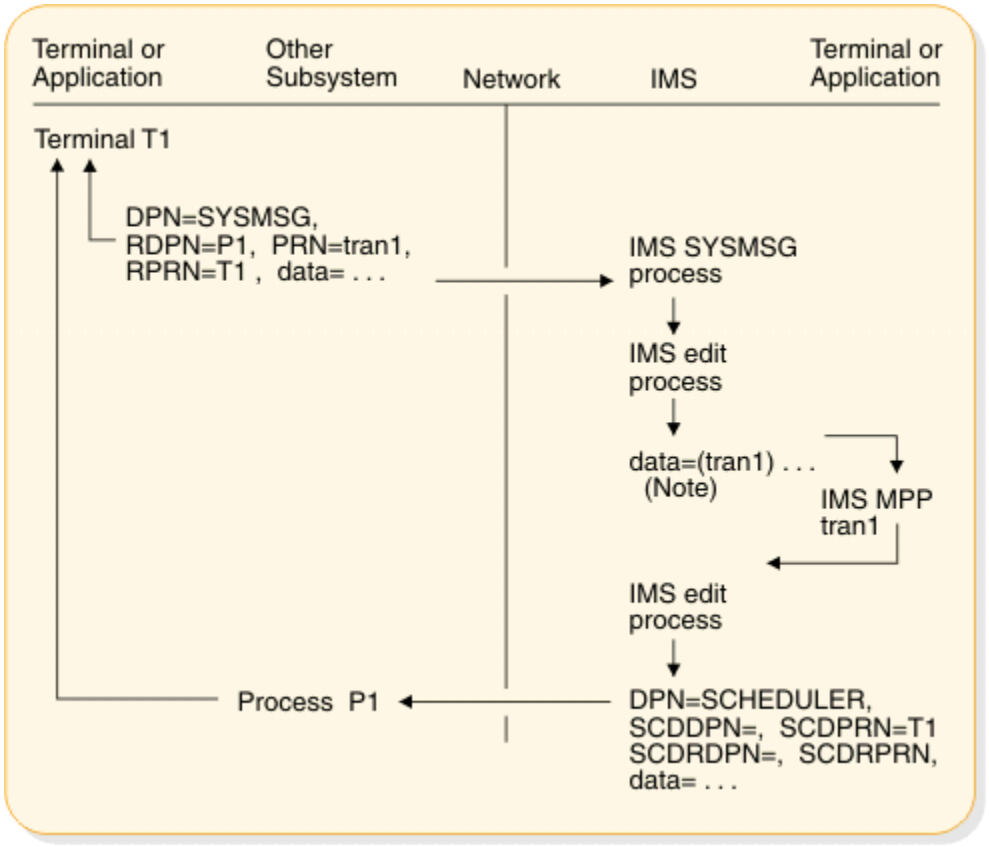


Figure 76. Example of SYSMSG with PRN from another subsystem terminal to IMS

Note: The PRN parameter becomes the IMS destination. The input RDPN and RPRN parameters become the output DPN and PRN parameters respectively for any resulting reply from the MPP.

ATTACH and SCHEDULER parameters with IMS MFS

The following series of figures illustrate the use of the ATTACH and SCHEDULER parameters when using IMS MFS for IMS input and output messages.

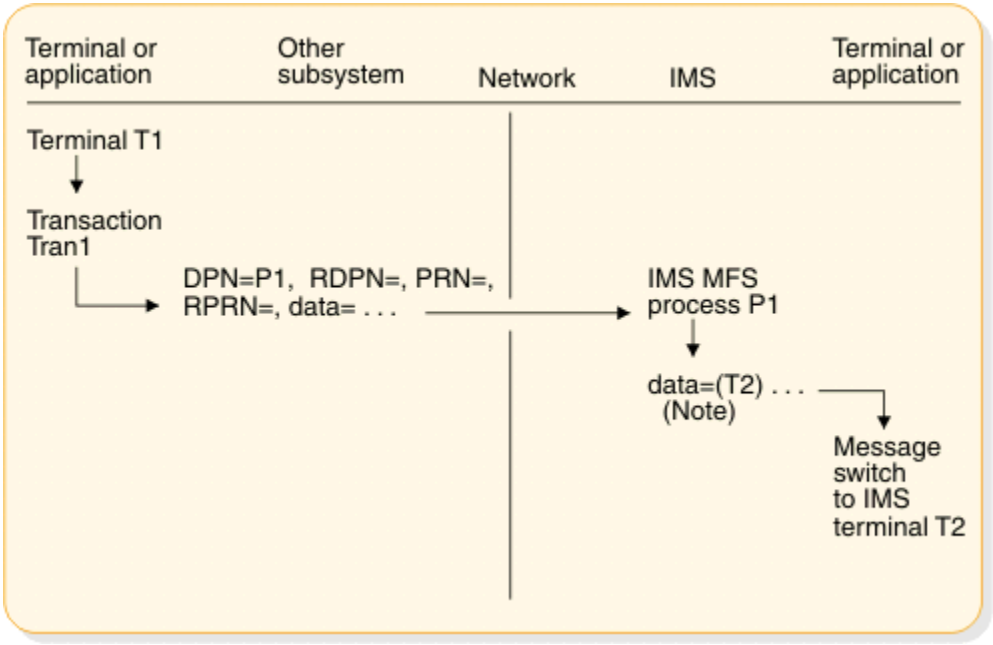


Figure 77. Example of message switch from other subsystem terminal to IMS terminal

Note: The IMS LTERM name for terminal T2 can be supplied as the input primary resource name (PRN), as a data field, or as an MFS-defined literal. RPRN and RDPN, if supplied, can be optionally included in the data presented to the IMS terminal by MFS. The DPN parameter must be supplied in the input ATTACH FM header to invoke MFS.

In the following figure, a transaction message originates from Terminal 1 and is sent to Process P1 on a non-IMS subsystem. The ATTACH parameters are then used to send the message to the MFS process P2 on the IMS subsystem. The MFS process P2 sends the message to Tran2 in the MPP region. The reply from the MPP region is sent back to the MFS process, which uses the ATTACH parameters to send the reply to the other subsystem. The other subsystem allocates Terminal 1 as a resource of Transaction Tran1 and the message is sent back to the terminal.

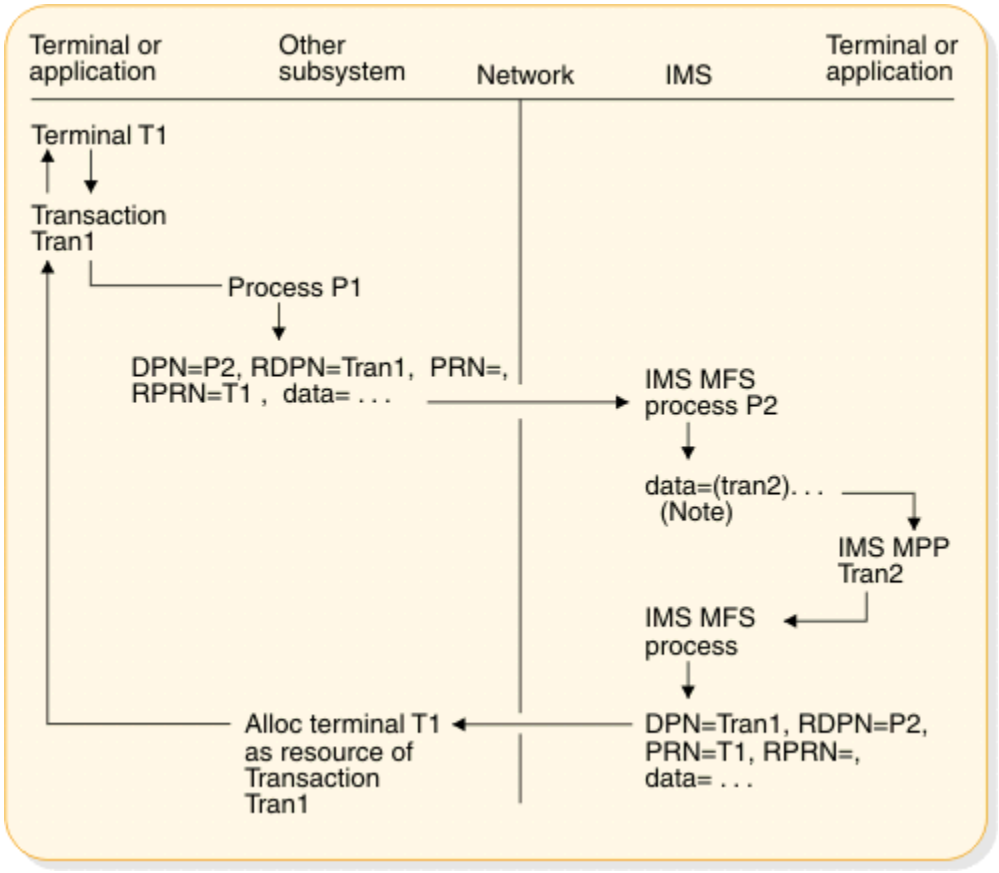


Figure 78. Example of other subsystem terminal accesses IMS application program with reply

Note: The IMS transaction code can be supplied as the input PRN, as a data field (first field for basic edit), or as an MFS-defined literal. The DPN parameter must be supplied to invoke IMS MFS DPM. If a reply inserted by the MPP is processed by MFS, the DPM process might create or override any of the output ATTACH parameters. The input RDPN and RPRN parameters become the output DPN and PRN parameters respectively for any resulting reply from the application program if they are not overridden by the DPM process.

In the following figure, a transaction message originates from Terminal 1 and is sent to Process P1 on a non-IMS subsystem. The ATTACH parameters are then used to send the message to the MFS DPM process P2 on the IMS subsystem. The MFS DPM process P2 sends the message data to Tran2 in the MPP region. The reply from the MPP region is sent back to the MFS DPM process, which uses the ATTACH parameters to send the reply to the non-IMS subsystem as Tran3, which is then stored for later retrieval by Terminal 1.

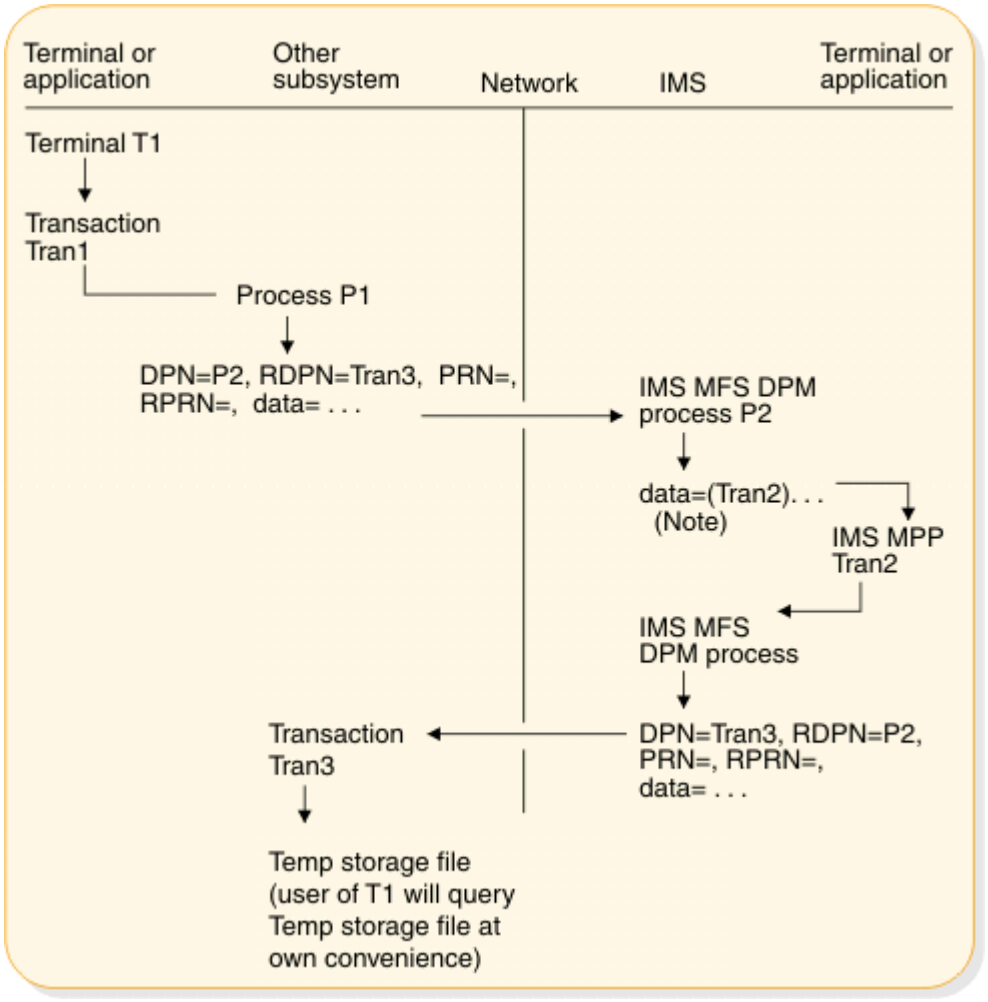


Figure 79. Example of IMS an MPP accessed from other subsystem terminal with reply to a temporary storage file of the other subsystem

Note: The IMS transaction code can be supplied as the input PRN, as a data field, or as an MFS-defined literal. The DPN parameter must be supplied to invoke IMS MFS DPM. If a reply is inserted by the MPP, the DPM process might override or create any of the output ATTACH parameters. The input RDPN and RPRN parameters become the output DPN and PRN parameters respectively for any resulting reply from the application program if they are not overridden by the DPM process.

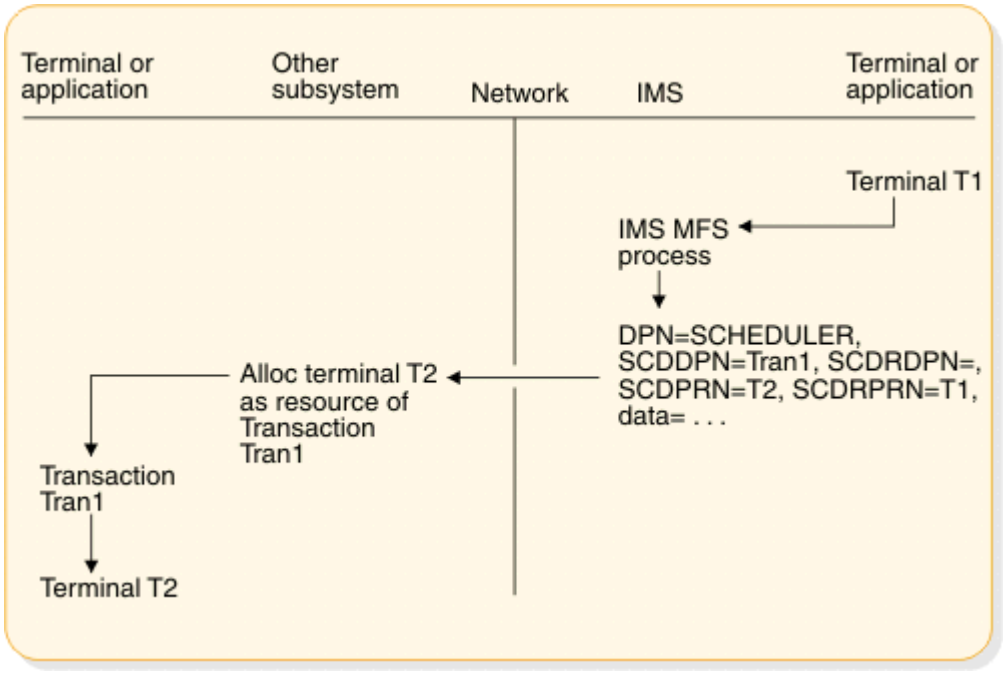


Figure 80. Example of a message switch from an IMS terminal to a terminal on another subsystem

MFS DPM is necessary to create the required output SCHEDULER parameters. The SCDDPN and SCDPRN can be entered as data from the terminal or application program. If a reply is returned to another IMS terminal or application program, the default SCDRPRN parameter can be overridden within the IMS output message's SCHEDULER FM header to specify the suggested IMS terminal or application to receive that reply. The reply is returned with this value in the SCDPRN field.

In the following figure, messages are routed from IMS Terminal 1 to the IMS MFS process either through a message switch or through an MPP or Fast Path message routing application. The MFS process P2 sends the message with SCHEDULER parameters to a transaction on another subsystem. The other subsystem then sends the reply message back using similar SCHEDULER parameters to IMS and the MFS process P2. The MFS edit process directs the message back to Terminal 1.

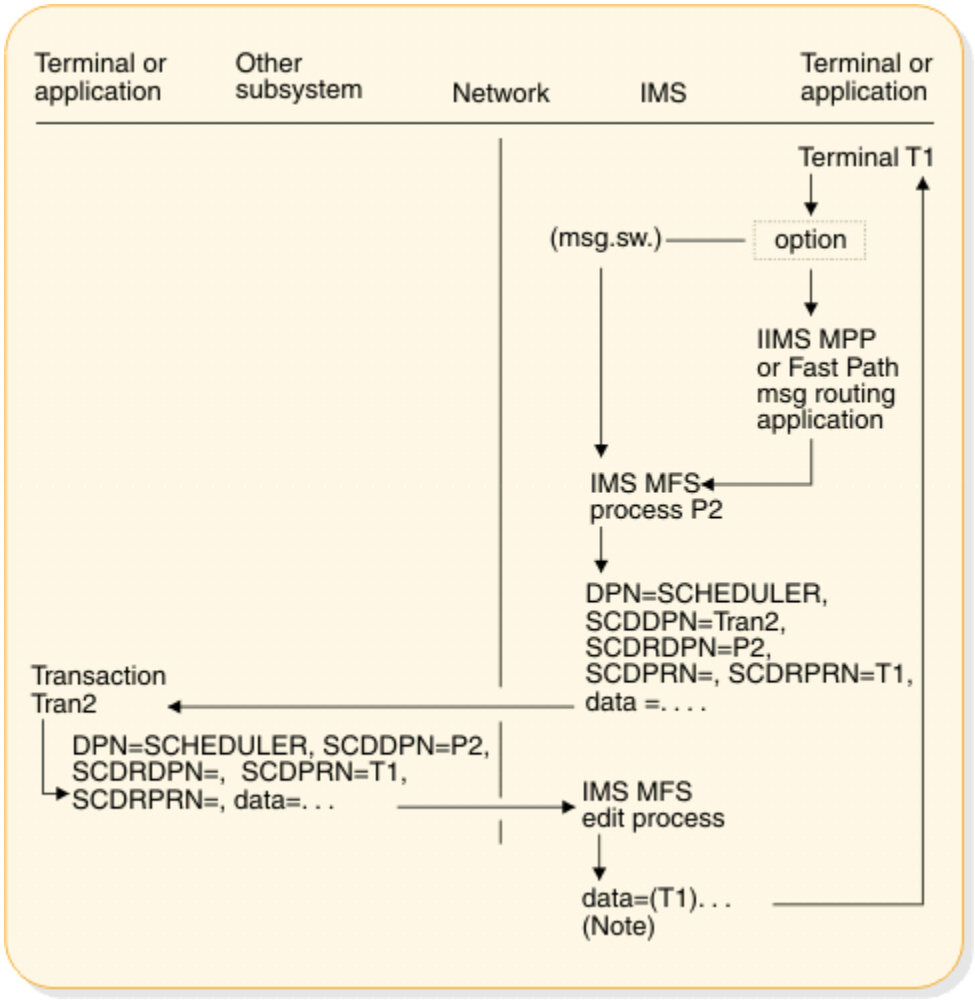


Figure 81. Example of message routing with IMS MFS

Note: The IMS LTERM name can be supplied as the input SCDPRN, as the first data field, or as an MFS-defined literal. The SCDDPN parameter must be supplied to invoke IMS MFS. MFS can override or create any of the output SCHEDULER parameters.

Chapter 34. How IMS and CICS use the ISC interface

When designing and implementing an ISC network that contains both IMS and CICS nodes you should be aware of the issues involved.

For ISC connections that are supported by VTAM, you should also be familiar with the basic system design concepts of both IMS and CICS running in a VTAM environment.

For ISC connections that are supported by TCP/IP, the TCP/IP support is provided by IMS Connect and most of the VTAM protocols do not apply. Communication between IMS and IMS Connect is enabled by the Structured Call Interface (SCI) of the IMS Common Service Layer (CSL).

The CICS information provided here is for planning purposes and should be verified in the CICS documentation or with a CICS system programmer.

Related reading: For more information on the design and coding of a CICS application that communicates with IMS through LU 6.1 ISC, see *CICS Transaction Server for z/OS CICS Intercommunication Guide*.

Functions available to the ISC session

The functions available to an ISC session between IMS and CICS differ depending on whether the connection is provided by TCP/IP or by VTAM and whether the processing is synchronous or asynchronous.

Related concepts

“Relationship of ISC and IMS execution modes” on page 461

Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

Overview of CICS synchronous and asynchronous processing for ISC

CICS implements ISC using its command level (EXEC) application programming interfaces (APIs). The CICS APIs can be synchronous or asynchronous.

The synchronous API uses the SEND and RECEIVE EXEC commands. In *synchronous mode* in an ISC session, the session is held between the moment a request is entered and the moment a reply is returned. Thus, a direct correlation can be made between the input request and the output reply.

ISC TCP/IP connections do not support the SEND and RECEIVE EXEC commands.

The asynchronous API uses the START and RETRIEVE EXEC commands. In *asynchronous mode* in an ISC session, no correlation can be made between an input request and an output reply. No assumptions can be made as to the timing of the output reply or the availability of any other output for that session.

CICS supports ISC TCP/IP sessions with only the START and RETRIEVE EXEC commands.

When used within the context of a CICS application program, the EXEC commands create the data flow control protocols for synchronous or asynchronous processing that are associated with messages on the session.

The CICS implementation is quite different from IMS, in which the data flow control protocols sent with a message are created by the system and based on:

- Whether the attributes (such as recoverability and segmentation) are defined for the message on the TRANSACT, TERMINAL, and SUBPOOL macro statements during IMS system definition or on an ETO logon descriptor
- For VTAM connections, whether the characteristics are defined in the bind parameters
- For TCP/IP connections, whether the characteristics are defined in the CICS IPCONN and TCPIPSERVICE resource definitions
- Whether the message is a reply that is returned on the same session as an associated request or on a different session than the associated request

© Copyright IBM Corp. 1974, 2020

- Whether the message is unsolicited asynchronous output

When an asynchronous message is sent from CICS to IMS, the sending transaction might terminate. The returned reply can be processed by a newly initiated CICS transaction according to parameters in the function management headers returned on the message by IMS.

The following figure illustrates the concept of synchronous SEND and RECEIVE processing.

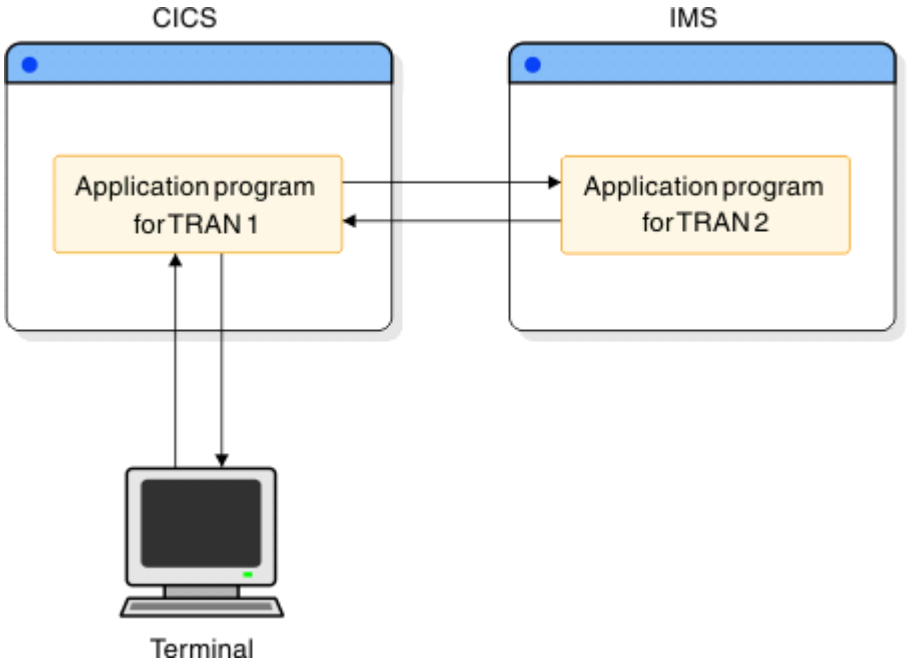


Figure 82. Synchronous processing with SEND and RECEIVE

The following figure illustrates the concept of asynchronous START and RETRIEVE processing.

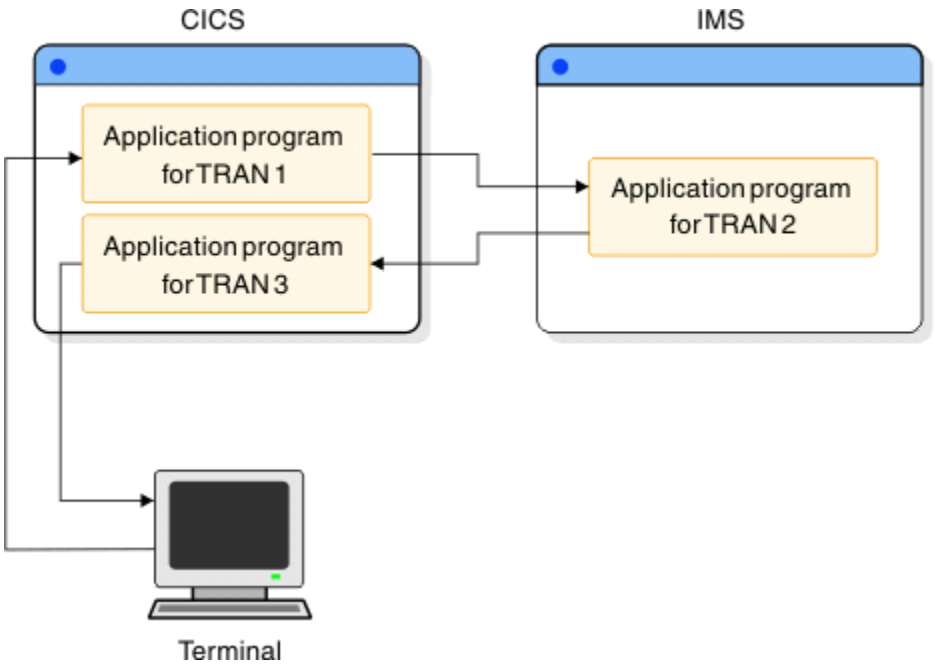


Figure 83. Asynchronous processing with START and RETRIEVE

Related concepts

CICS: CICS-to-IMS applications

Functions available to an ISC TCP/IP session

CICS supports ISC TCP/IP sessions with a private IP interconnectivity (IPIC) protocol and an asynchronous command level (EXEC) application programming interface (API).

The asynchronous API provided by CICS for ISC TCP/IP sessions uses the **START** and **RETRIEVE** EXEC commands. When used within the context of a CICS application program, the **START** and **RETRIEVE** EXEC commands create the IPIC data flow control protocols for the asynchronous processing that is associated with messages on the session.

A **START** command causes an IPIC header to be issued with the message. Messages sent to IMS with the **START** command can be only one chain in length and are sent with BB/EB. CICS uses the **RETRIEVE** command to obtain asynchronous messages from the session for CICS transaction processing. The other IPIC data flow control protocols issued with an asynchronous message depend upon the type of message issued.

The use of the CICS asynchronous **START** and **RETRIEVE** EXEC commands with ISC TCP/IP sessions precludes the use of any function for which CICS requires a synchronous flow. For example, the **START** and **RETRIEVE** EXEC commands support only single-chain messages. Additionally, certain functions that are supported by ISC VTAM sessions are not supported by ISC TCP/IP sessions.

The following tables show the functions available on ISC TCP/IP sessions with CICS.

The tables do not take into consideration the recovery aspects of any of the listed combinations. A "Yes" indicates only that the approach is possible. However, a system designer should take into account the need for ease of recovery and restart.

Table 121. Functions available to an ISC TCP/IP session with CICS: CICS front end

Functions	ISC TCP/IP support
Nonresponse or nonconversational mode IMS input transaction	Yes
Nonresponse or nonconversational mode IMS output transaction	Yes
Response mode IMS transaction (including Fast Path)	No
Conversational mode nonlast IMS input transaction	No
Conversational mode last IMS input transaction	No
Recoverable"1" on page 590 IMS transaction	N/A
Nonrecoverable"1" on page 590 IMS transaction	N/A
IMS message switch	Yes
Front-End Switch (FES) exit routine (DFSFEBJ0)	No
IMS operator commands	**/DISPLAY**and **/RDISPLAY** only
Message Format Service (MFS)"2" on page 590	No
Single-segment input/output IMS transaction	Yes
Multi-segment input/output IMS transaction	No

Table 122. Functions available to an ISC TCP/IP session for CICS EXEC commands: IMS front end

Functions	ISC TCP/IP support
CICS transaction	Yes
CICS operator command (system transaction)	No
MFS"2" on page 590	No

Notes for CICS EXEC tables:

1. A transaction is defined as recoverable or nonrecoverable during IMS system definition and is acceptable to IMS only when the sync point protocols and transaction definitions are consistent. See "Relationship of ISC and IMS execution modes" on page 461 and "Keeping half sessions synchronized" on page 506 for more information.
2. MFS maps data between the session and the IMS application program. In CICS, basic mapping support (BMS) maps data at the request of the CICS application program. BMS does not interact with the session, but can be used to handle mapping of data to and from IMS. IMS MFS and CICS BMS do not communicate.

Related tasks

"ISC support for TCP/IP" on page 452
TCP/IP can be used to support ISC communications between IMS and CICS subsystems.

Functions available to an ISC VTAM session

ISC sessions with CICS on VTAM connections support both the synchronous and asynchronous CICS APIs.

This topic identifies synchronous and asynchronous command sequences available to a CICS application on ISC VTAM sessions.

The support for front-end/back-end system utilization that the IMS Front-End Switch exit routine provides uses ISC in asynchronous mode.

CICS **SEND** and **RECEIVE** commands create the predefined SNA protocols associated with synchronous processing when **SEND** is used with the INVITE option. **SEND** INVITE causes an SNA ATTACH function management header and change-direction to be issued with the message. The message is sent from CICS to the remote IMS subsystem for processing. When the message reply is returned, it is processed by a **RECEIVE** command issued within the same CICS transaction that issued the first message.

Exception: Restarted transactions are handled differently.

When the **SEND** command is used with the LAST option, it causes the message to be sent on the session with an ATTACH function management header and carries both begin-bracket and end-bracket (BB/EB) status. IMS treats this special case of synchronous processing in the same way as an IMS asynchronous transaction is treated. If IMS generates an output reply as a result of an input generated with **SEND** LAST, the returned reply can be processed by a newly initiated CICS transaction. The IMS output reply is also sent with ATTACH BB/EB. CICS uses **RECEIVE** to process that reply.

The **START** and **RETRIEVE** commands create the SNA protocols associated with asynchronous processing. A **START** command causes an SNA ATTACH function management header and a concatenated SCHEDULER function management header to be issued with the message. Messages sent to IMS with the **START** command can be only one chain in length and are sent with BB/EB. CICS uses the **RETRIEVE** command to obtain asynchronous messages from the session for CICS transaction processing. The other SNA data flow control protocols issued with an asynchronous message depend upon the type of message issued.

The following series of tables summarize the functions available to an ISC VTAM session between IMS and CICS. You must know whether the flow on a session is to be synchronous or asynchronous in order to determine what functions are available to that session. See the notes following the tables for further explanations.

The tables do not take into consideration the recovery aspects of any of the listed combinations. A "Yes" indicates only that the approach is possible. However, a system designer should take into account the need for ease of recovery and restart.

Table 123. Functions available to an ISC VTAM session for CICS EXEC commands: CICS front end

Functions available	SEND(INVITE)/ RECEIVE (Synchronous)	SEND(LAST) (Synchronous)	START/ RETRIEVE"1" on page 592 (Asynchronous)
Nonresponse or nonconversational mode IMS input transaction	No	Yes	Yes
Nonresponse or nonconversational mode IMS output transaction	Yes"2" on page 592	N/A	Yes
Response mode IMS transaction (including Fast Path)	Yes"3" on page 592	No	No
Conversational mode nonlast IMS input transaction	yes	No"4" on page 592	No
Conversational mode last IMS input transaction	No	No"4" on page 592	Yes
IMS message switch	No	Yes	Yes
IMS operator command	Yes"5" on page 592	Yes"6" on page 592	Yes"6" on page 592
Recoverable"7" on page 592 IMS transaction	N/A	N/A	N/A
Nonrecoverable"7" on page 592 IMS transaction	N/A	N/A	N/A
Single-segment input/output IMS transaction	Yes	Yes	Yes
Multi-segment input/output IMS transaction	Yes	Yes	No

Table 124. Functions available to an ISC VTAM session for CICS EXEC commands: IMS front end

Functions available	SEND(INVITE)/ RECEIVE (Synchronous)	SEND(LAST) (Synchronous)	START/ RETRIEVE"1" on page 592 (Asynchronous)
CICS transaction	No	No	Yes
CICS operator command (system transaction)	No	No	No

Table 125. Functions available to an ISC VTAM session for CICS EXEC commands: MFS and BMS support

Functions Available"8" on page 592	SEND(INVITE)/ RECEIVE (Synchronous)	SEND(LAST) (Synchronous)	START/ RETRIEVE"1" on page 592 (Asynchronous)
Without IMS MFS	Yes	Yes	Yes
MFS without paging	Yes	Yes	Yes
MFS autopaged input	Yes"9" on page 592	Yes"9" on page 592	No

Table 125. Functions available to an ISC VTAM session for CICS EXEC commands: MFS and BMS support (continued)

Functions Available[“8” on page 592]	SEND(INVITE)/ RECEIVE (Synchronous)	SEND(LAST) (Synchronous)	START/ RETRIEVE[“1” on page 592] (Asynchronous)
MFS autopaged output	Yes[“10” on page 592, “2” on page 592]	N/A	No
MFS demand-paged output	Yes	N/A	No[“11” on page 592]
BMS support[“8” on page 592]	N/A	N/A	N/A

Notes for CICS EXEC tables:

1. Single-chain messages only.
2. The nonresponse-mode reply to a transaction sent with SEND LAST is sent ATTACH BB/EB. CICS uses RECEIVE (without SEND) to obtain the message from the session. (See “CICS to IMS using the SEND LAST EXEC command” on page 607.)
3. IMS forces response mode regardless of system definition if a transaction is specified externally (using the function management header) as synchronous. See “Relationship of ISC and IMS execution modes” on page 461.
4. IMS conversational mode requires that IMS terminate the conversation by sending the last reply with EB. The exception to this is explained in “CICS versus IMS conversation mode” on page 625.
5. The replies that result from the IMS **/DISPLAY**, **/FORMAT**, and **/RDISPLAY** commands are sent asynchronously after IMS replies synchronously to the input command.
6. Supported only for **/DISPLAY**, **/RDISPLAY**, and **/FORMAT**.
7. A transaction is defined as recoverable or nonrecoverable during IMS system definition and is acceptable to IMS only when the sync point protocols and transaction definitions are consistent. See “Relationship of ISC and IMS execution modes” on page 461 and “Keeping half sessions synchronized” on page 506 for more information.
8. MFS maps data between the session and the IMS application program. BMS maps data at the request of the CICS application program. BMS does not interact with the session, but can be used to handle mapping of data to and from IMS. IMS's MFS and CICS's BMS do not communicate.
9. Both single- and multiple-message chains can be used, but only one chain can occur per SEND.
10. To avoid tying up the session, when processing synchronous autopaged output from IMS, CICS should read all pages of the IMS output before processing it.
11. Although demand paging is possible between IMS and CICS, it is not recommended, because it requires complex CICS application coding and a complex terminal user interface. A preferable approach is to create autopaged output from IMS to CICS and use CICS page retrieval in the local system for paging.

Related concepts

“Keeping half sessions synchronized” on page 506
Sync-point responses (DR2) are used between ISC session partners to ensure that both partners' sync-point managers can commit or back out recoverable resources synchronously.

“CICS to IMS using the SEND LAST EXEC command” on page 607
In a system in which CICS is the front-end subsystem and IMS the back-end subsystem, IMS supports the use of the CICS synchronous API to allow access to IMS asynchronous transactions. SEND LAST

is used to process IMS nonresponse mode and nonconversational transactions, message switches, and the **/DISPLAY**, **/RDISPLAY** and **/FORMAT** commands.

Related reference

Front-End Switch exit routine (DFSFEBJ0) (Exit Routines)

ISC communication with CICS over TCP/IP

TCP/IP can be used to support ISC connections between IMS and IBM CICS Transaction Server for z/OS subsystems.

ISC supports TCP/IP only for connections between IMS and CICS. ISC TCP/IP support uses a private protocol, IP interconnectivity (IPIC), that is defined by CICS. The use of IPIC is generally consistent with the protocols that are defined by SNA for ISC VTAM and is transparent to application programs that use ISC.

IMS Connect provides the TCP/IP connection support for ISC. The Structured Call Interface (SCI) of the IMS Common Service Layer (CSL) provides the communication path between IMS and IMS Connect.

Both dynamically and statically defined ISC terminals can use TCP/IP. However, if you are defining new ISC terminals to support TCP/IP, use dynamic terminals. Dynamic terminals do not require a system definition or a cold start of IMS and provide much greater flexibility than static ISC terminals. Dynamic ISC terminals are defined by coding logon descriptors in either the DFSDSCMx or the DFSDSCTy members of the IMS.PROCLIB data set.

Related tasks

"IMS Connect and TCP/IP communications" on page 139
The IMS Connect function of IMS provides access to both IMS DB and IMS TM from TCP/IP-enabled environments.

Overview of ISC TCP/IP support

ISC TCP/IP connections use the CICS IP interconnectivity protocol for session control and data flow control between IMS and CICS subsystems. IMS Connect provides TCP/IP support.

ISC TCP/IP does not use SNA VTAM or the protocols, commands, headers, and so forth, that are used with VTAM. However, for consistency, some VTAM terms are used in an ISC TCP/IP context where the meaning is the same.

Otherwise, from an operational, application programming, and high-level configuration perspective, ISC TCP/IP communication and ISC VTAM communication are generally the same.

You use the same IMS commands to control the ISC sessions from IMS. IMS applications programs do not need to be sensitive to the use of either ISC TCP/IP or ISC VTAM. Statically defined ISC LU 6.1 terminals are defined with the same IMS stage-1 system definition macros, and dynamically defined terminals use similar ETO logon descriptors.

Statically defined ISC LU 6.1 terminals and dynamically defined ISC nodes can use TCP/IP to connect to CICS subsystems.

If you use dynamically defined terminals, the Extended Terminal Option (ETO) is required.

The following figure shows the main components that support ISC TCP/IP communication between IMS and CICS: the Structured Call Interface (SCI) for communication between IMS and IMS Connect, and IMS Connect for communication with CICS over TCP/IP.

The figure compares these components with the main components that support ISC VTAM communication. The primary difference between the two communication types is that for ISC TCP/IP the path between IMS and CICS is provided by SCI, IMS Connect, and TCP/IP. For ISC VTAM, the path between IMS and CICS is provided by VTAM only.

Both communication types support input that originates from either an IMS user or a CICS user. In either case, the input is sent to the partner subsystem, which processes the transaction and returns a reply.

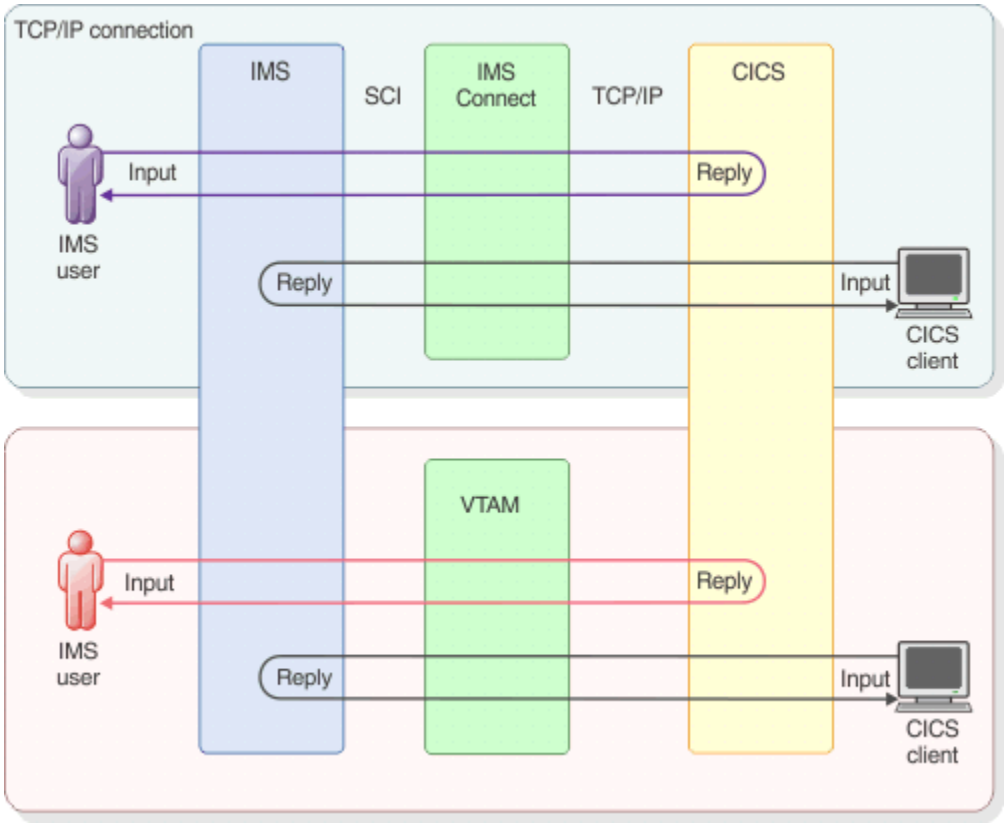


Figure 84. Comparison of ISC TCP/IP and ISC VTAM communication flows

Related tasks

"Extended Terminal Option (ETO)" on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

"IMS Connect and TCP/IP communications" on page 139
The IMS Connect function of IMS provides access to both IMS DB and IMS TM from TCP/IP-enabled environments.

Requirements of ISC TCP/IP support

The requirements for the ISC TCP/IP function include requirements for CICS, ETO, IMS Connect, and a minimal CSL.

- IBM CICS Transaction Server for z/OS, Version 5.1 (or later) must be used.
- IMS Connect is required to provide TCP/IP socket connection support for IMS.
- The Common Service Layer (CSL) with at least the Structured Call Interface (SCI) and the Operations Manager (OM) is required. SCI is required for communications between IMS and IMS Connect and OM is required for type-2 command support.
- A single point of control (SPOC) program, such as the IMS TSO SPOC, must be used to issue type-2 commands to the OM API or REXX SPOC API.
- For each IMS subsystem that uses dynamically defined terminals with ISC TCP/IP, the IMS Extended Terminal Option (ETO) is required.

Related concepts

Common Service Layer overview (System Administration)

A single point of control (SPOC) program in CSL (System Administration)

Related tasks

"IMS Connect and TCP/IP communications" on page 139

The IMS Connect function of IMS provides access to both IMS DB and IMS TM from TCP/IP-enabled environments.

"Extended Terminal Option (ETO)" on page 59
These topics introduce the extended terminal option (ETO) and include an overview of ETO and the information you need to administer ETO terminals in an IMS TM Network.

Restrictions for ISC TCP/IP support

The following functionality is not supported by ISC TCP/IP.

- CICS transactions that use the SEND(INVITE)/RECV protocol for synchronous communication
- CICS transactions that use the SEND(LAST)/RECV protocol
- Extended Recovery Facility (XRF)
- IMS operator commands except for /DISPLAY and /RDISPLAY
- IMS conversational mode transaction
- IMS response mode transactions, including Fast Path
- IMSplex Terminal Management (STM)
- Front-End Switch (FES)
- Message Format Service (MFS)
- VTAM Generic Resources (VGR)

Security for ISC TCP/IP connections

ISC TCP/IP connections with CICS can be secured by authentication of the user name or subpool name, transaction authorization, the Signon exit routine, a security product, such as RACF, and message encryption.

When a parallel session is initiated, IMS authenticates the user ID that is associated with the initiation request. IMS derives the user ID from the value of the USER keyword on the **/OPNDST** command or, for statically defined terminals, from the value of the NAME parameter on the SUBPOOL macro. Passwords are not used.

For each transaction sent on the connection, IMS checks the authority of the user ID to schedule the transaction.

For statically defined ISC TCP/IP terminals, IMS authenticates the user ID during session initiation and checks the transaction authority of the user ID only if the AUTOSIGN option is enabled.

For IMS, message encryption on ISC TCP/IP connections can be provided by the Application Transparent Transport Layer Security (AT-TLS) of the z/OS TCP/IP stack. AT-TLS encrypts and decrypts messages independently from IMS Connect. The messages that are sent and received by IMS Connect are unencrypted. Because IMS Connect uses separate send and receive sockets for a parallel session, your AT-TLS administrator must configure AT-TLS to support both sockets.

You can also use AT-TLS to perform client authentication on ISC TCP/IP connection requests that are received by IMS. CICS must be configured to provide a security certificate when initiating an ISC TCP/IP connection. If a certificate is not provided with a connection request, AT-TLS issues a message and rejects the connection request. IMS Connect issues error message HWSV5000E.

The configuration of AT-TLS encryption and client authentication support in IMS must be coordinated with the configuration of SSL support in CICS. In CICS, SSL support is enabled for ISC TCP/IP connections by implementing IPIC bind time security in the IPCONN and TCPIPSERVICE resource definitions.

Restriction: ISC TCP/IP connections with CICS do not support the IMS **/SIGN** command.

For more information about AT-TLS, see the *z/OS Communications Server: IP Configuration Guide*

Related reference
/OPNDST command (Commands)

SUBPOOL macro (System Definition)

Setting up an ISC TCP/IP connection with CICS

Setting up an ISC TCP/IP connection with CICS requires steps in IMS and IMS Connect, as well as coordination with the CICS administrators.

Prerequisites: Before you can define an ISC TCP/IP connection with CICS, the following features must be enabled in IMS:

- The Structured Call Interface (SCI) and Operations Manager (OM) components of the Common Service Layer (CSL)
- If you are using dynamically defined ISC terminals or nodes, the Extended Terminal Option (ETO)

The following topics describe how to define an ISC TCP/IP connection with CICS.

Defining terminals for ISC TCP/IP connections

Although both dynamically defined and statically defined ISC terminals can use TCP/IP, if you are defining new terminals, use dynamic terminals. Dynamic terminals do not require a system definition or a cold start of IMS and provide greater flexibility than static ISC terminals.

Dynamic ISC terminals are defined by coding logon descriptors in either the DFSDSCMx or the DFSDSCTy members of the IMS.PROCLIB data set.

Static and dynamic ISC TCP/IP terminals are equivalent in many aspects:

- The connection can be initiated or terminated via IMS commands.
- Session status can be displayed via IMS commands.
- Input and output messages can be transported via either static or dynamic terminal.

An important aspect in which static and dynamic terminals differ is that static terminals must be defined with IMS stage 1 system definition macros and require a restart of IMS, whereas dynamic terminals are created during runtime as needed by using terminal attributes that are defined by Extended Terminal Option (ETO) logon descriptor in the IMS.PROCLIB data set.

Defining dynamic terminals for ISC TCP/IP sessions

When the IMS Extended Terminal Option (ETO) is enabled, you can use ETO logon descriptors to enable the dynamic creation of terminals for ISC TCP/IP LU 6.1 sessions.

At least one logon descriptor must be defined to enable IMS to create ISC TCP/IP terminals dynamically. At least one logon descriptor must be defined for each IMS Connect instance that supports dynamically defined ISC TCP/IP nodes.

Logon descriptors are stored in two members in the IMS.PROCLIB data set:

DFSDSCM*x*
For the default logon descriptors that are created by IMS during a cold start.

DFSDSCT*y*
For the logon descriptors that are created by your installation.

Recommendation: When you create your own logon descriptors, to preserve them across stage-1 system definition processing, store your logon descriptors in the DFSDSCT*y* PROCLIB member.

You can add to or modify the descriptors in the DFSDSCM*x* member. However, during the processing of a stage-1 system definition, IMS deletes the contents of the DFSDSCMx member and re-creates all of the default descriptors.

IMS creates a default logon descriptor for ISC TCP/IP terminals named DFSTCP. To use the default ISC TCP/IP descriptor for sessions started in IMS, you must specify it explicitly by using the LOGOND keyword in the /OPNDST command. ISC TCP/IP sessions that are initiated from CICS can use the default logon descriptor, but only when no other logon descriptor matches the node name specified by CICS.

1. To create a logon descriptor for an ISC TCP/IP terminal, code an ETO logon descriptor that specifies UNITYPE=ISCTCPIP and LCLICON=*lcl_imsconnect_name*.

   The IMS Connect instance that is specified on the LCLICON keyword provides the TCP/IP support, including the network address and port of the CICS subsystem.
2. Optional: define an ETO user descriptor.

For example:

```
L ISCTCP1  UNITYPE=ISCTCPIP LCLICON=HWS1
L ISCTCP2  UNITYPE=ISCTCPIP LCLICON=HWS1
L ISCTCP3  UNITYPE=ISCTCPIP LCLICON=HWS1
L ISCTCP4  UNITYPE=ISCTCPIP LCLICON=HWS1
L ISCTCP5  UNITYPE=ISCTCPIP LCLICON=HWS1
L TERMA    UNITYPE=3270 UNIT=3284
L DFS327P  UNITYPE=3270 UNIT=3284
U DFSUSER  OPTIONS=(TRANRESP)
```

Related concepts

“ETO descriptors” on page 63
A *descriptor* provides information to IMS when IMS builds a dynamic resource for a logon or a signon. The four types of ETO descriptors are: logon descriptors, user descriptors, MSC descriptors, and MFS device descriptors.

Logon descriptors (System Definition)

ETO descriptors (System Definition)

ETO descriptor overrides (System Definition)

Related tasks

“Creating logon descriptors” on page 81
Logon descriptors provide IMS with information about the physical characteristics of the terminals that establish logon sessions. These characteristics must be consistent with the VTAM logon BIND characteristics.

“Creating user descriptors” on page 84
User descriptors provide information relating to user options and user structure names. IMS needs user descriptors in order to create control blocks that enable users to use ETO terminals.

Configuring statically defined LU 6.1 ISC terminals to use TCP/IP

Configuring a statically defined ISC terminal to use TCP/IP requires a cold start of IMS, updates to stage 1 system definition macros, and the DFSDCxxx PROCLIB member.

By default, statically defined ISC terminals use VTAM for communication. The ISCTCPIP parameter in the DFSDCxxx PROCLIB member enables TCP/IP for statically defined ISC terminals.

Requirements: To support TCP/IP, ISC must be defined to support parallel sessions. ISC TCP/IP communications does not support single-session ISC terminals.

Statically defined ISC terminals must specify UNITYPE=LUTYPE6 in the TYPE stage-1 system definition macro.

To configure a statically defined LU 6.1 ISC terminal to use TCP/IP:

1. If you are switching an existing terminal from VTAM to TCP/IP, ensure that UNITTYPE=LUTYPE6 is specified on the TYPE macro of the system definition macros that define the terminal.

   If UNITTYPE=LUTYPE6 is not specified, an error message will be issued when the ISCTCPIP parameter is processed in the DFSDCxxx member and TCP/IP will not be enabled for the terminal.
2. If you are defining a new terminal, define the terminal by coding the appropriate system definition macros.

   The TYPE macro must specify UNITYPE=LUTYPE6 and parallel session must be defined by coding the VTAMPOOL and SUBPOOL macros. For example:

```
TYPE  UNITYPE=LUTYPE6
TERMINAL  NAME=LU6NDPA,MSGDEL=SYSINFO,EDIT=(NO,NO),     X
SESSION=3,OPTIONS=NOMTOMSG,                             X
```

```
COMPT1=(SINGLE1,VLVB),                                  X
COMPT2=(SINGLE2,VLVB),                                  X
COMPT3=MULT1
VTAMPOOL
     SUBPOOL  NAME=LU6SPA,MSGDEL=SYSINFO
        NAME  LU6LTPA1,COMPT=2,ICOMPT=1
     SUBPOOL  NAME=LU6SPB
        NAME  LU6LTPB1
     SUBPOOL  NAME=LU6SPC
        NAME  LU6LTPC1
```

If UNITTYPE=LUTYPE6 is not specified, when the ISCTCPIP parameter is processed in the DFSDCxxx member, IMS will issue an error message and TCP/IP will not be enabled for the terminal.

3. In the DFSDCxxx PROCLIB member, define the terminal to use TCP/IP communications by specifying the name of the terminal and the name of IMS Connect instance that provides TCP/IP support on the ISCTCPIP keyword, as in the following example: `ISCTCPIP=(LU6NDPA,ICON1)`
4. Cold start IMS.

   IMS does not read changes to the ISCTCPIP keyword during warm or emergency restarts.

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

Macros used in IMS environments (System Definition)

Falling back from TCP/IP to VTAM for static ISC TCP/IP terminals

Falling back from TCP/IP to VTAM for a static LU 6.1 ISC terminal requires updates to the DFSDCxxx PROCLIB member and a cold start of IMS.

The default communication protocol for LU 6.1 ISC terminals is VTAM.

To fall back to VTAM:

1. Delete the ISCTCPIP parameter that specifies the node name of the terminal from the DFSDCxxx PROCLIB member.
2. Cold start IMS.

   If a warm or emergency restart is performed, the ISCTCPIP parameter remains in effect.

Related reference

CSLDCxxx member of the IMS PROCLIB data set (System Definition)

Defining IMS Connect support for ISC TCP/IP links

IMS Connect support for ISC TCP/IP is defined by the ISC, RMTCICS, and TCPIP statements in the IMS Connect configuration member in the IMS.PROCLIB data set.

The ISC statement defines most of the attributes of an ISC TCP/IP link to IMS Connect.

The RMTCICS statement defines the network addressing information of the CICS subsystem.

The TCPIP statement specifies the ports on which IMS Connect receives transactions and data replies from CICS.

You can define one or more ISC TCP/IP links between an IMS subsystem and a CICS subsystem.

ISC links that connect to the same CICS subsystem can use the same RMTCICS connection definition or they can use separately defined RMTCICS connections:

- ISC links that use the same RMTCICS connection definition must specify the same value on the RMTCICS parameter.
- ISC links that use separately defined RMTCICS connections to the same CICS subsystem must specify different IDs on the RMTCICS parameter; however the different RMTCICS statements that each ISC statement references must specify the same host name or IP address and port number.

To define IMS Connect support for ISC TCP/IP code the following statements:

- The ISC statement.

To code an ISC statement, you need the following values:

  - The IMS ID of the local IMS system
  - The APPLID of the remote CICS subsystem
  - The node name that defines the ISC link to IMS
  - The names by which IMS and IMS Connect are known to IMSplex
  - The CICSPORT number of the port on which IMS Connect receives transactions and data replies from the CICS subsystem
  - The ID of the RMTCICS statement that defines the network address and port of the CICS subsystem
- The RMTCICS statement.

  To code an RMTCICS statement, you need the following values:

  - The host name of the CICS subsystem. This value is also specified on the CICSPORT keyword in the TCPIP statement.
  - The port number of the port on which the CICS subsystem receives transactions and data replies from IMS.
- The TCPIP statement.

  To add ISC TCPIP support to an existing TCPIP statement, specify the CICSPORT parameter to define the port on which IMS Connect will receive transactions and data replies from CICS. This value is also specified on the CICSPORT keyword in the ISC statement.

The following example shows an IMS Connect configuration member that provides IMS Connect support for ISC TCP/IP. The HWS statement is included for completeness.

```
HWS=(ID=HWS1,XIBAREA=100,RACF=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(9998,19998,LOCAL),RACFID=GOFISHIN,
     TIMEOUT=000,
     IPV6=Y,
     PORT=(ID=15554),
     CICSPORT=(ID=1111,KEEPAV=1000),
     CICSPORT=(ID=3333,KEEPAV=1000),
     CICSPORT=(ID=6666,KEEPAV=1000),
     CICSPORT=(ID=7777,KEEPAV=1000),
     EXIT=(HWSSMPL0,HWSSMPL1,HWSCSLO0,HWSCSLO1,HWSSOAP1))
ISC=(ID=CICSA1,NODE=LU6NDPA,
     IMSPLEX=(MEMBER=HWS1,TMEMBER=PLEX1),
     LCLIMS=IMS1,RMTCICS=CICS1,CICSAPPL=CICS1,CICSPORT=1111)
ISC=(ID=CICSA2,NODE=LU6NDPB,
     IMSPLEX=(MEMBER=HWS1,TMEMBER=PLEX1),
     LCLIMS=IMS1,RMTCICS=CICS1,CICSAPPL=CICS1,CICSPORT=3333)
ISC=(ID=CICSA5,NODE=ISCTCP1,
     IMSPLEX=(MEMBER=HWS1,TMEMBER=PLEX1),
     LCLIMS=IMS1,RMTCICS=CICS1,CICSAPPL=CICS1,CICSPORT=6666)
ISC=(ID=CICSA6,NODE=ISCTCP2,
     IMSPLEX=(MEMBER=HWS1,TMEMBER=PLEX1),
     LCLIMS=IMS1,RMTCICS=CICS1,CICSAPPL=CICS1,CICSPORT=7777)
RMTCICS=(ID=CICS1,HOSTNAME=ABC.EXAMPLE.COM,PORT=23456)
```

Related reference

RMTCICS statement (System Definition)

ISC statement (System Definition)

TCPIP statement (System Definition)

Defining an ISC TCP/IP link in CICS

An ISC TCP/IP link is configured in CICS as an IP interconnectivity (IPIC) connection by two CICS resource definitions: an IPCONN resource and a TCPIPSERVICE resource.

The CICS IPCONN resource defines the attributes that are required for CICS to send messages on an IPIC connection. From the CICS perspective, the IPCONN statement defines the outbound attributes of an IPIC connection.

One IPCONN resource is required for each parallel session. In CICS, the IPCONN resources can be created dynamically with the CICS autoinstall program or you can define them yourself.

In the IPCONN resource definition, the following parameters must match values that are specified in IMS:

- IPCONN(*ipconnname*) must match either the user name that is specified on the IMS /OPNDST command or, for statically defined ISC TCP/IP terminals, the NAME keyword on a SUBPOOL IMS system definition macro.
- HOST must match the host name of IMS Connect that is specified on the HOSTNAME keyword in the TCPIP configuration statement in the IMS Connect HWSCFGxx member of the IMS.PROCLIB data set..
- PORT must match the value that is specified on the CICSPORT parameter in both the RMTCICS and TCPIP configuration statements in the IMS Connect HWSCFGxx member of the IMS.PROCLIB data set. IMS Connect receives messages from CICS on this port.

The CICS TCPIPSERVICE resource defines the attributes that are required for CICS to receive messages on an IPIC connection. From the CICS perspective, the TCPIPSERVICE resource defines the inbound attributes of an IPIC connection.

In the TCPIPSERVICE resource definition, the following parameters must match values that are specified in the IMS:

- HOST must match the host name of IMS Connect that is specified on the HOSTNAME keyword of the TCPIP configuration statement.
- PORT must match the value that is specified to IMS Connect on the PORT parameter of the ISC configuration statement.

The following example shows the definitions for the CICS IPCONN and TCPIPSERVICE resources.

```
DELETE GROUP(TCPIPGRP)
DEFINE TCPIPSERVICE(TSA)
       PORT(23456)
       PROTOCOL(IPIC)
       GROUP(TCPIPGRP)

DEFINE IPCONN(LU6SPA)
       APPLID(LU6SPA)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(1111)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)

DEFINE IPCONN(LU6SPB)
       APPLID(LU6SPB)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(3333)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)

DEFINE IPCONN(LU6SPC)
       APPLID(LU6SPC)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(3333)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)

DEFINE IPCONN(USER01)
       APPLID(USER01)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(6666)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)
```

```
DEFINE IPCONN(USER02)
       APPLID(USER02)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(6666)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)

DEFINE IPCONN(USER03)
       APPLID(USER03)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(7777)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)

DEFINE IPCONN(USER04)
       APPLID(USER04)
       AUTOCONNECT(YES)
       HOST(XYZ.EXAMPLE.COM)
       PORT(7777)
       SENDCOUNT(5)
       RECEIVECOUNT(5)
       TCPIPSERVICE(TSA)
       GROUP(TCPIPGRP)

DEFINE TRANSACTION(SR1A)
       PROGRAM(IMSSRT1A)
       GROUP(DFHPPTDR)

DEFINE PROGRAM(IMSSRT1A)
       GROUP(DFHPPTDR)
       LANGUAGE(COBOL)

ADD GROUP(TCPIPGRP) LIST(DRVRLIST)
```

Starting a session with CICS on an ISC TCP/IP link

An ISC parallel session on an ISC TCP/IP link can be started from IMS or from the partner CICS system.

Starting a parallel session on an ISC TCP/IP link from IMS

From IMS, you start an ISC TCP/IP parallel session with CICS by issuing the **/OPNDST NODE** command.

Prerequisites:

- Before you can start an ISC TCP/IP session with CICS, the connection must be defined in IMS, IMS Connect, and CICS as described in “Setting up an ISC TCP/IP connection with CICS” on page 596.
- In IMS, the ISC terminal must be defined to use TCP/IP. If the terminal is not defined to use TCP/IP, IMS attempts to start the session by using VTAM.

You need the following information to start an ISC parallel session with CICS over TCP/IP:

- The node name that identifies the target CICS subsystem.
- The user name of the parallel session.
- For sessions that use dynamically defined terminals, the name of the ETO logon descriptor that defines the terminal attributes of the connection and identifies the IMS Connect instance that provides TCP/IP support.

You can find the node name on the NODE keyword of the ISC statement that defines the target CICS subsystem to IMS Connect.

For sessions that use statically defined terminals, the node name is also defined to IMS on the ISCTCPIP keyword of the DFSDC*xxx* PROCLIB member, which defines the terminal as using TCP/IP, as well as the NAME keyword of the TERMINAL system definition macro.

The requirements for specifying the user name of a parallel session differ depending on whether the parallel sessions and the ISC terminals are statically or dynamically defined.

If either IMS or CICS uses static definitions, the user name that you specify on the **/OPNDST NODE** command must match the corresponding value in the static definitions. In IMS, the user name of a statically defined parallel session is defined by the NAME keyword of the SUBPOOL system definition macro. In CICS, the user name is defined by either the IPCONN name or the APPLID keyword of the IPCONN resource definition.

If IMS dynamically defines the ISC terminals, the user name is specified only on the USER keyword of the **/OPNDST NODE** command in the IMS system.

If the CICS autoinstall function is active in the CICS subsystem and the parallel session is started from IMS, CICS uses the user name defined in IMS to name the IPCONN resource. However, if the user name is greater than four characters in length, CICS uses only the last four characters.

The user name specified in the USER keyword of the **/OPNDST NODE** command can be up to 8 characters. However, CICS uses only the last four characters of the user name if the IPCONN autoinstall program is active in CICS.

ETO logon descriptors define the attributes of the dynamic terminals that are used for the ISC connection to CICS, including the name of the IMS Connect instance that provides TCP/IP support.

Specifying a logon descriptor on the **/OPNDST NODE** command is optional. If the LOGOND keyword is not specified and a parallel session is not already open, the value of the NODE keyword is used to search for a logon descriptor. Alternatively, the name of the logon descriptor can be provided by a Logon exit routine (DFSLGNX0).

If a parallel session is already open on a node and a different logon descriptor is specified than was used to start the first parallel session, the **/OPNDST NODE** command is rejected.

If the logon descriptor that is specified on the **/OPNDST NODE** command does not specify TCP/IP support, IMS attempts to open the session as a VTAM node.

The **/OPNDST NODE** command is the only way to start an ISC TCP/IP parallel session with CICS from IMS.

Restriction: ISC supports only parallel sessions on TCP/IP connections.

You can start an ISC parallel session with CICS by issuing the **/OPNDST NODE** command.

If the session starts successfully, IMS issues message DFS2064I to the master terminal and IMS and CICS perform a *capability exchange* to validate the session and server attributes.

Related concepts

“ETO descriptors” on page 63
A *descriptor* provides information to IMS when IMS builds a dynamic resource for a logon or a signon. The four types of ETO descriptors are: logon descriptors, user descriptors, MSC descriptors, and MFS device descriptors.

Logon descriptors (System Definition)

Related tasks

“Creating logon descriptors” on page 81
Logon descriptors provide IMS with information about the physical characteristics of the terminals that establish logon sessions. These characteristics must be consistent with the VTAM logon BIND characteristics.

Related reference

/OPNDST command (Commands)

Starting a parallel session on an ISC TCP/IP link from CICS

From IBM CICS Transaction Server for z/OS, you start an ISC TCP/IP parallel session with IMS by issuing the CICS command `SET IPCONN(`*`ipconnnm`*`) ACQUIRE`.

Prerequisites: Before you can start an ISC TCP/IP session with IMS from CICS:

- The connection must be defined in IMS, IMS Connect, and CICS as described in "Setting up an ISC TCP/IP connection with CICS" on page 596.
- In IMS, the ISC terminal must be defined to use TCP/IP. If the terminal is not defined in IMS to use TCP/IP, IMS rejects the request to start the session.

To start an ISC TCP/IP session from CICS, you need the 1-to-4 character IPCONN name from the IPCONN resource that defines the ISC connection in CICS. This value is the same as the USER value that identifies the parallel session in IMS.

If you migrated LUTYPE 6.1 links from ISC VTAM to ISC TCP/IP, the IPCONN name is likely the same as the name of the CONNECTION resource that defined the ISC VTAM link.

You can issue the SET IPCONN ACQUIRE command through several different CICS interfaces, including:

- The CICS Explorer®
- The CICS system programming interface (SPI)
- The CICS master terminal transaction CEMT
- The CICS command-level interpreter (CECI) transaction

For example, to start an ISC parallel session with IMS from CICS by using CEMT, issue the following command:

When a session starts successfully, IMS issues message DFS2064I to the master terminal and IMS and CICS perform a *capability exchange* to validate the session and server attributes.

To check that the session was started, issue the INQUIRE IPCONN(*ipconnnm*) command by using any of the CICS command interfaces.

After the session starts, CICS application programs can use the CICS START and RETRIEVE commands to process IMS nonresponse mode and nonconversational transactions, message switches, and the IMS / DISPLAY, /RDISPLAY, and /SIGN operator commands.

Related reference

CICS Transaction Server for z/OS

Terminating an ISC TCP/IP session

Terminating an ISC session that uses TCP/IP is generally the same as terminating an ISC session that uses VTAM.

You can terminate ISC TCP/IP session in an orderly manner, where work in progress is finished before termination completes, or you can terminate ISC TCP/IP sessions unconditionally, where termination is immediate and work in progress cannot be finished before termination completes.

When sessions terminate abnormally, work in progress usually cannot be finished before termination completes.

Related reference

/CLSDST command (Commands)
/QUIESCE command (Commands)
/STOP NODE command (Commands)
/CHECKPOINT command (Commands)

Terminating an ISC TCP/IP session in an orderly manner

During an orderly termination, both IMS and the partner subsystem complete normal processing before the session is terminated. No work is left pending and the connection is terminated in a cold state.

When the connection is restarted, IMS and the partner subsystem do not need to resynchronize the connection.

- To terminate an ISC TCP/IP session without disrupting work that is already in progress, issue the **/QUIESCE** IMS type-1 command.

  The IMS **/CHECKPOINT** command with the **FREEZE**, **PURGE**, or **DUMPQ** parameter and the **QUIESCE** parameter will also initiate an orderly termination. The **QUIESCE** parameter ensures that message queues are emptied before the session is terminated. When all terminals indicate that shutdown is complete, IMS completes checkpoint processing.

Terminating an ISC TCP/IP session unconditionally

When an ISC TCP/IP session is terminated unconditionally, any work that is in progress at the time of termination, is not completed before the session is terminated. When the connection is restarted, IMS and the partner subsystem must resynchronize the connection before any pending work can be finished.

The IMS MTO can terminate the network immediately during the processing of an orderly session termination by issuing an IMS **/CLSDST**, **/STOP**, or **/CHECKPOINT** command. The **/STOP** command leaves the session in a STOPPED state.

Terminating the session by using an IMS Connect command also shuts down the session immediately.

- To terminate an ISC TCP/IP session immediately, issue the **/CLSDST** IMS type-1 command, which terminates all processing immediately.

Abnormal termination of an ISC TCP/IP session

Abnormal termination of an ISC TCP/IP session can occur as a result of transmission or protocol errors, or errors in data that make that data unacceptable to the receiving message processing program.

Because an ISC session involves two peer-level systems, error recovery processing and abnormal session termination processes can differ depending on which system initiated the connection.

Any work in progress at the time the of the abnormal termination is not completed before the session is terminated. When the connection is restarted, IMS and the partner subsystem must resynchronize the connection before any pending work can be finished.

Restarting an ISC TCP/IP session

You restart an ISC session that uses TCP/IP by issuing the /OPNDST command.

The CICS AUTOCONNECT function only acquires a connection automatically when CICS and IMS communicate for the first time. If you shut down a CICS region and restart it, CICS recovers the IPCONN resources from the local CICS catalog and does not automatically reacquire any ISC TCP/IP session. You must issue IMS /OPNDST commands for both statically defined terminals and dynamically defined nodes to re-establish communications again. The CICS AUTOCONNECT facility is set on the IPCONN statement within the CICS DFHCSD data set that contains resource definitions.

If the ISC TCP/IP connection was stopped in IMS Connect, be sure to restart the connection in IMS Connect before restarting ISC communication in IMS. In IMS Connect, the communication can be stopped on the ISC connection between IMS and IMS Connect, on the TCP/IP connection between IMS Connect and CICS, or both.

CICS front-end transaction types supported by ISC over TCP/IP

As a front-end system, CICS supports only the asynchronous START and RETRIEVE interface for ISC TCP/IP connections with IMS.

Consequently, CICS application programs must use the IMS non-response or non-conversational mode transactions. CICS application programs cannot use either the synchronous or asynchronous SEND and RECEIVE interfaces with TCP/IP. To use the CICS asynchronous SEND and RECEIVE interface or CICS synchronous distributed transaction processing (DTP) with ISC, the ISC connections must use VTAM.

When IMS is the front-end system, the transaction flow from IMS to CICS on an ISC TCP/IP connection is also asynchronous, the same as it is on an ISC VTAM connection.

General flow of CICS EXEC commands within a CICS application

The design of a CICS application program using the CICS EXEC command level application programming interfaces is determined by whether the transaction being sent on the ISC session is to be processed using SEND/RECEIVE, SEND LAST, or START/RETRIEVE.

The sequence of EXEC commands issued is determined by whether the transaction is defined as recoverable or nonrecoverable. Program design can also depend on whether CICS is the front-end system (initiating a transaction) or back-end system (replying to an IMS transaction).

This topic presents an overview of synchronous and asynchronous transaction processing flow within CICS. Understanding the content and functions of the ATTACH and SCHEDULER FM headers is helpful in understanding this topic.

Related concepts

"Functions available to the ISC session" on page 587
The functions available to an ISC session between IMS and CICS differ depending on whether the connection is provided by TCP/IP or by VTAM and whether the processing is synchronous or asynchronous.

Related reference

"Coding function management headers for CICS" on page 629
CICS uses some of the same SNA-defined function management header fields that are used by IMS.

CICS to IMS using SEND/RECEIVE EXEC commands

In a system in which CICS is the front-end subsystem and IMS the back-end subsystem, the **SEND/RECEIVE** commands are used to process IMS response mode (including Fast Path) and conversational transactions, and IMS commands.

The general CICS application program flow for a synchronous message from CICS to IMS is shown in the following figure.

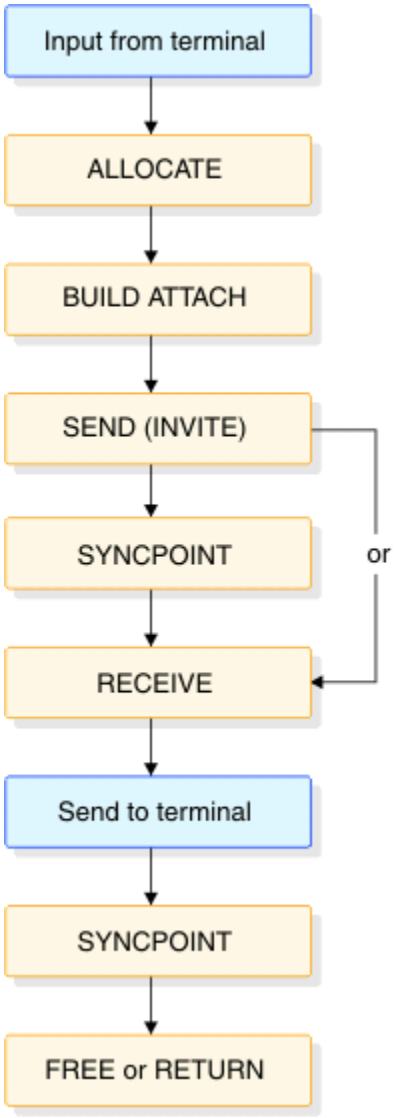


Figure 85. Application program flow using SEND/RECEIVE from CICS to IMS

In this example, the CICS application reads the input message from the terminal and establishes a CICS-to-IMS session using the **ALLOCATE** command.

The CICS application program builds the required ATTACH function management header (the only header required, because this is a synchronous application) by using the **BUILD ATTACH** EXEC command. The RPROCESS and RRESOURCE fields should be specified in the **BUILD ATTACH** to provide for restart. Issuing **SEND** causes the output message to be constructed. **SEND INVITE** causes change-direction (CD) to be appended to the output message.

The output message is not sent across the session until the next command is executed. The next EXEC command that is issued depends upon the definition of the transaction within IMS, because this command must both append the sync-point request to the message as required and send the message across the ISC session. If this transaction is defined to IMS as recoverable, the next command must be **SYNCPOINT**. **SYNCPOINT** causes the outbound message to be sent with an RQD2. If the transaction is defined as nonrecoverable, a **SYNCPOINT** or a **RECEIVE** command can be issued next. A **RECEIVE** command immediately following a **SEND** command (without an intervening **SYNCPOINT** command) causes the outbound message to be issued with RQE1.

RECEIVE issued after either the **SYNCPOINT** or **SEND** commands reads the IMS reply. The reply from IMS is returned synchronously on the session (with the ATTACH FM header). If the transaction reply is to a response mode (including Fast Path) transaction, a command (except **/DISPLAY**, **/RDISPLAY**, and **/FORMAT**), or the last conversational reply, it is returned to CICS carrying RQD2 EB if recoverable, or RQD1

EB if nonrecoverable. If the transaction is a nonlast conversational reply, it is returned to CICS carrying RQE2 and CD (because the conversation continues).

If the transaction is defined within IMS as a nonrecoverable-inquiry, the **CONVERSE** command can be used. The **CONVERSE** command acts as if a **SEND** command were issued, immediately followed by a **RECEIVE** command. Transactions sent with a **CONVERSE** command carry BB/CD and a request for an exception response (RQE1). If transactions other than those defined as nonrecoverable are sent using CONVERSE, an error results.

After issuing RECEIVE, the application program must save and check the values in the EXEC interface block (EIB) before performing any additional processing. If the transaction reply is received successfully as determined by checking the EIB, the application now issues a sync point (SYNCPOINT). Issuing this sync point causes a DR2 response to be returned to the IMS transaction reply. If CICS wants to perform any application processing based upon the contents of the IMS reply message, including sending an output message to the terminal, that processing is performed before the sync point is issued. This ensures that CICS resources and IMS resources are committed in synchronism.

If the transaction completes successfully, the application program must free the session using either a **FREE** or **RETURN EXEC** command.

Related concepts

“Recovery and restart concepts” on page 635
This topic describes the system and user functions that must be performed to recover an ISC session after a session, system, or application failure and assumes the reader understands the role of the STSN command in session resynchronization.

Related reference

“Coding function management headers for CICS” on page 629
CICS uses some of the same SNA-defined function management header fields that are used by IMS.

CICS to IMS using the SEND LAST EXEC command

In a system in which CICS is the front-end subsystem and IMS the back-end subsystem, IMS supports the use of the CICS synchronous API to allow access to IMS asynchronous transactions. SEND LAST is used to process IMS nonresponse mode and nonconversational transactions, message switches, and the **/DISPLAY**, **/RDISPLAY** and **/FORMAT** commands.

The general CICS application program flow for CICS to IMS using SEND LAST is shown in the following figure.

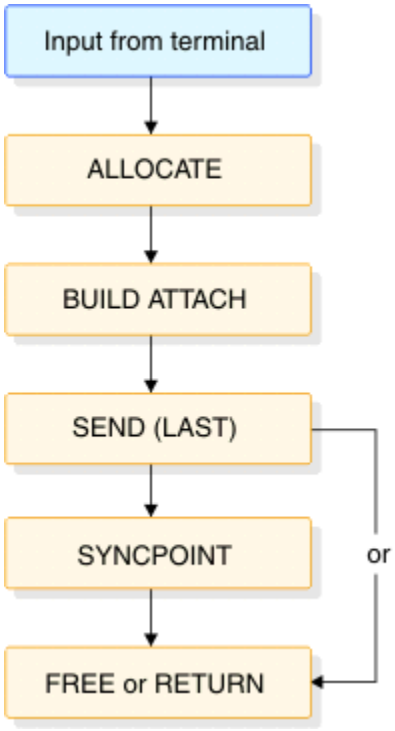


Figure 86. Application program flow using SEND LAST from CICS to IMS

In this example, the CICS application reads the input message from the terminal and establishes a CICS-to-IMS session using the ALLOCATE command.

The CICS application program builds the required ATTACH function management header (the only header required, because this is a synchronous application) by using the BUILD ATTACH EXEC command. The RPROCESS and RRESOURCE fields should be specified in the BUILD ATTACH to identify the terminal and transaction to be used to process the reply. Issuing a SEND command causes the output message to be constructed. A SEND LAST command causes end-bracket to be appended to the message.

The output message is not sent on the session until the next command is executed. The next EXEC command that is issued depends upon the definition of the transaction within IMS, because this command both appends the sync-point request to the message and issues the message on the ISC session. If this transaction is defined to IMS as recoverable, the next command must be SYNCPOINT. SYNCPOINT causes the outbound message to be issued with an RQD2. If the transaction is defined as nonrecoverable, either SYNCPOINT, FREE, or RETURN can be issued next. FREE or RETURN issued immediately after the SEND and without an intervening SYNCPOINT causes the outbound message to be issued with RQE1.

If IMS generates a reply message as a result of receiving this transaction, that reply from IMS is returned with the ATTACH FM header. In order to get the IMS reply message from the session, CICS must issue a RECEIVE (because the reply message carries only ATTACH).

Related reference

"Coding function management headers for CICS" on page 629
CICS uses some of the same SNA-defined function management header fields that are used by IMS.

IMS to CICS using the RECEIVE EXEC command

When IMS has a reply to send to CICS that results from a message sent by CICS with SEND LAST (ATTACH BB/EB), IMS sends that message using the ATTACH function management header and BB/EB. In order to receive the message from the session, CICS must initiate a new transaction (or a new instance of the original transaction) that uses the RECEIVE EXEC command for this purpose.

The general CICS application program flow for this is shown in the following figure.

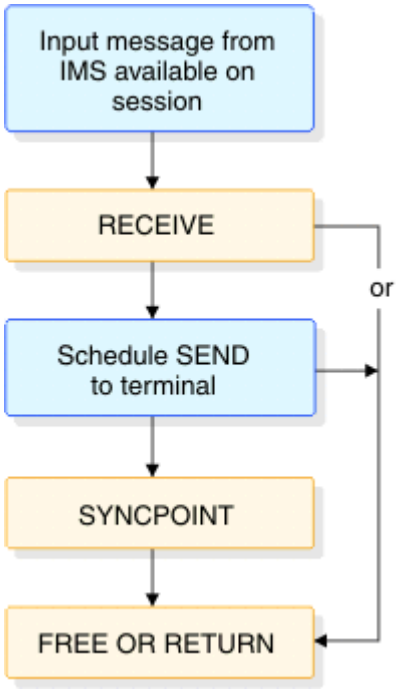


Figure 87. Application program flow using RECEIVE from IMS to CICS

The reply from IMS is returned synchronously on the session (with the ATTACH FM header) and carries BB/EB. A recoverable reply carries RQD2; a nonrecoverable reply carries RQD1. The RPROCESS (RDPN) and RRESOURCE (RPRN) fields sent to IMS are automatically wrapped by IMS into the DPN and PRN fields of the outbound ATTACH FMH.

CICS examines the returned reply and uses the DPN, or the first four characters of the data, to determine the transaction that can process the returned reply. RECEIVE is used to obtain the IMS reply from the session. The PRN field is passed to the initiated transaction, indicating the terminal to which the returned reply is to be sent.

After issuing RECEIVE, the application program must save and check the values in the EXEC interface block (EIB) before performing any additional processing.

CICS must schedule an asynchronous transaction to send the returned reply to the terminal by issuing a START command. The transaction that is initiated is brought up, and owns the terminal to which the reply is to be sent. This asynchronous transaction uses the START interface to communicate with the terminal, and all processing within this transaction occurs asynchronously to the transaction that initiated it. If the transaction that sends output to the terminal is scheduled successfully, the originating application now issues a sync point (SYNCPOINT), if required. Issuing this sync point causes an appropriate response to be returned to the IMS transaction reply. If the transaction is not scheduled successfully, issuing the sync point causes an appropriate exception response to be returned to the IMS transaction reply. If CICS wants to perform any application processing based upon the contents of the IMS reply message, that processing is performed before the sync point is issued. This ensures that CICS resources and IMS resources are committed in synchronism.

If the processing completes successfully, the application program must free the session using either a FREE or RETURN EXEC command.

Coding asynchronous messages

When receiving asynchronous messages (sent with ATTACH and SCHEDULER function management headers) from IMS, CICS invokes a CICS-supplied *mirror transaction* to obtain these incoming messages from the session.

The mirror transaction is CICS's name for the SCHEDULER process. This mirror transaction examines the incoming data stream and schedules (using START) the transaction that is to be initiated as a result of the

incoming DPN or the first four characters of the data. The receiving CICS transaction is assumed to be one of the following:

- ISC edit (ISCE), the default DPN set by IMS
- The transaction whose identifier was placed in the DPN field of the ATTACH FM header by IMS's wrapping of the incoming RDPN (the RTRANSID specified on the CICS START)
- A transaction whose identifier has been supplied by an MFS MOD

CICS uses the PRN in the incoming message as the TERMID parameter of the START command so that the transaction is initiated and owns the CICS terminal (if applicable) to which this message should be written.

A SYNCPOINT command is now issued to return the appropriate response to IMS and to complete the scheduling of the asynchronous transaction. Following the sync point, a RETURN command is issued to terminate the transaction and free the session. The response returned to IMS causes the output asynchronous message to be dequeued from the IMS output queue.

CICS to IMS using the START/RETRIEVE EXEC commands

CICS uses **START/RETRIEVE** commands to process IMS nonresponse mode and nonconversational transactions, message switches, and the **/DISPLAY**, **/RDISPLAY**, and **/FORMAT** commands.

The general CICS application program flow for an asynchronous message from CICS to IMS is shown in the following figure.

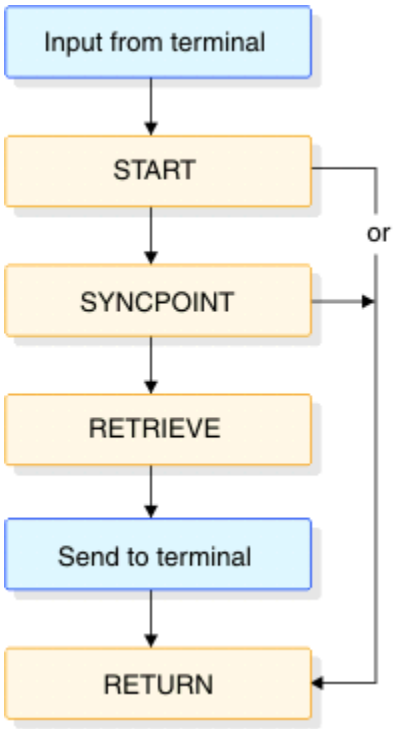


Figure 88. Application program flow using START/RETRIEVE from CICS to IMS

After the input message is received from the terminal, the transaction is assembled using the **START** command. The values to be contained in the ATTACH and SCHEDULER function management header DPN, PRN, RDPN, and RPRN fields are specified as parameters on the **START** command. The DPN and PRN are used by IMS to determine the editing process to be initiated and the destination of the incoming message. The RDPN and RPRN are used to identify the CICS terminal and transaction to which the reply is to be returned. The **START** command must also specify **NOCHECK** if this is an IMS nonrecoverable transaction, and **NOCHECK PROTECT** if this is an IMS recoverable transaction. (**PROTECT** can optionally be used for nonrecoverable transactions.)

NOCHECK is a mandatory parameter for an ISC session with IMS. In a CICS-CICS ISC session, the use of **START** causes a return code reply to be sent to the initiating CICS from the receiving CICS. This return code indicates that the receiving CICS has scheduled the transaction to be executed as a result of the message it has received. This return code does not occur on an IMS-CICS session. **NOCHECK** informs the sending CICS that no such response is to be expected. **START NOCHECK** creates a message carrying BB/EB RQE1.

PROTECT, specified for recoverable transactions, delays the sending of the transaction on the session until CICS successfully takes a SYNCPOINT. Specifying **START** causes begin- and end-bracket (BB/EB) and the ATTACH and SCHEDULER function management headers to be appended to the outbound message. **NOCHECK PROTECT** requests a definite response (RQD2).

A **SYNCPOINT** command is now issued to append the appropriate sync point protocols and to send the message on the session. If this transaction elects to wait for a reply, it issues a **RETRIEVE** command. If this transaction does not choose to wait for a reply, it can terminate by issuing **RETURN** following either the **START** command or the **SYNCPOINT** command. The CICS mirror transaction schedules a new instance of this transaction when a reply is received.

If the transaction chooses to wait for the reply, it can issue **RETRIEVE** with or without the WAIT parameter. Issuing **RETRIEVE** without the **WAIT** parameter causes CICS to check for any queued asynchronous input for this transaction and terminal. If such input is immediately available, it satisfies the **RETRIEVE** command. If no such input is immediately available, an appropriate indication is returned to the **RETRIEVE** command, and the application can then perform other processing or issue RETURN to terminate.

If **RETRIEVE WAIT** is issued, CICS does not return control to the application program until an asynchronous message is sent from IMS destined for this transaction and this terminal. Using **WAIT** causes the terminal be held until an IMS reply is received for it.

In both cases (**RETRIEVE** with or without **WAIT**), when a message is sent from CICS to IMS requesting asynchronous processing, no assumptions can be made by the originating application as to the timing of the output reply or the availability of any other output. That is, if unsolicited asynchronous output is pending for CICS, this output can be sent to CICS before the reply to this asynchronous transaction is returned. Therefore, requests and replies are not correlated within CICS in an asynchronous environment. Issuing **RETRIEVE** obtains any reply from IMS on any session that has the indicated transaction and terminal identification.

IMS to CICS using the RETRIEVE EXEC command

The following figure illustrates the general CICS application program flow in asynchronous mode from IMS to CICS.

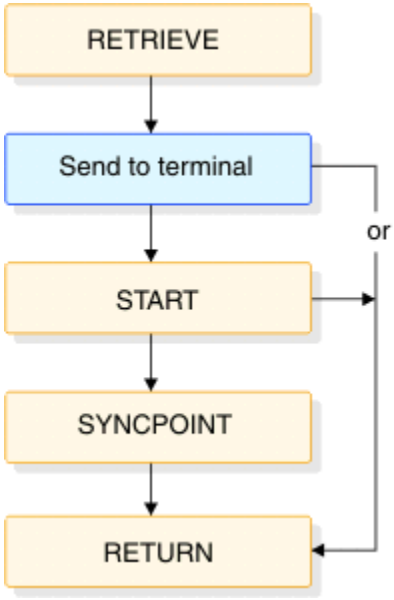


Figure 89. Application program flow using RETRIEVE from IMS to CICS

Replies resulting from previous asynchronous input transactions and unsolicited asynchronous output from an IMS front-end to a CICS back-end are sent to CICS asynchronously (that is, with the ATTACH and SCHEDULER FM headers). When IMS is the front-end subsystem, asynchronous mode is the only flow supported. The CICS mirror transaction obtains the input, analyzes the incoming message, and schedules (using START) the transaction that is to be initiated as a result of the incoming DPN or the first four characters of the data. If this transaction is sending output to a terminal, the mirror transaction issues the START with the TERMID parameter (from the PRN parameter of the incoming FMH), so that the asynchronous transaction that is scheduled owns the terminal to which output is to be sent.

If a reply to IMS is required, the CICS application must wrap the incoming RTERMID and RTRANSID into the TERMID and TRANSID fields to be used in a subsequent START. These fields indicate, respectively, the destination LTERM or transaction within IMS and the IMS editor or MFS MID that is to receive the message. The START command to send this reply to IMS is processed.

If no reply to IMS is required, RETURN can be issued to terminate the session.

Related concepts

"CICS to IMS using the START/RETRIEVE EXEC commands" on page 610
CICS uses **START/RETRIEVE** commands to process IMS nonresponse mode and nonconversational transactions, message switches, and the **/DISPLAY**, **/RDISPLAY**, and **/FORMAT** commands.

Related reference

"Coding function management headers for CICS" on page 629
CICS uses some of the same SNA-defined function management header fields that are used by IMS.

Commands that should not be used on an ISC session

Do not use the **WAIT SIGNAL** or **WAIT TERMINAL** commands on an ISC session.

The **WAIT SIGNAL** command cannot be used in a CICS-IMS session.

The **WAIT TERMINAL** command is not normally used in a CICS-IMS ISC session.

Selecting appropriate CICS installation options for ISC

This topic describes only those system definition options and resource definition options that have unique considerations for an IMS-CICS ISC environment.

Some of the parameters coded during the CICS installation process must be compatible with some of the IMS system definition parameters coded on the IMS system definition macros. The topic "Defining compatible IMS and CICS nodes" on page 614 brings together the definitions provided here with the IMS system definition information provided in "Statically defining an ISC node to IMS" on page 480.

Coding CICS system definition options

The CICS standard pregenerated system as supplied on the distribution tape includes a pregenerated version for each of the CICS management modules required to support an Intersystem Communication environment that includes all of the required system definition options.

Related reading:

- For information on system definition, see *CICS Transaction Server for z/OS CICS Intercommunication Guide* for the specifications necessary to generate a CICS subsystem that participates in IMS-CICS Intersystem Communication.
- For a checklist of recovery options, see *CICS Transaction Server for z/OS Recovery and Restart Guide*.

Preparing CICS resource definition

CICS user-created tables and Resource Definition Online (RDO) describe the database and data communications environment, and the treatment of the elements in that environment.

They contain information about terminals, files, databases, programs, transactions, transient data destinations, and temporary storage data identifiers. They are created independently of system definition, but must be present for the system to be operational. The CICS System Log and Dynamic Log, which are optional in some CICS environments, are required for IMS-CICS ISC.

Related reading:

- For more information on table preparation and RDO options, see the *CICS Transaction Server for z/OS CICS Resource Definition Guide*.
- For a summary of the parameters required for IMS-CICS Intersystem Communication, see *CICS Transaction Server for z/OS CICS Intercommunication Guide*.
- For a checklist of recovery-related options, see *CICS Transaction Server for z/OS Recovery and Restart Guide*.

Defining IMS-CICS LU 6.1 links

A CICS link to an IMS system requires a definition of the connection (or system) and a separate definition of each of the sessions.

Resource Definition Online or Macro-Level Resource Definition can be used to define an ISC link to IMS. IMS-CICS ISC links are implemented with LU 6.1 protocol.

The following table shows the RDO form of definition for individual LU 6.1 sessions, and shows the macro form for how the operands are related.

The TRMTYPE, TRMIDNT, SYSIDNT, NETNAMQ, and SESTYPE operands must be coded for each session that you define. The remaining operands of TYPE=TERMINAL can optionally be coded on the TYPE=SYSTEM macro to provide defaults for all the defined sessions. Also, the CONNECT, DATASTR, and RECFM operands of TYPE=SYSTEM can be coded for individual sessions, if required.

Table 126. Defining an LU 6.1 link with individual sessions

RDO definition	Macro-level definition

`DEFINE`
`  CONNECTION(sysidnt)`
`  GROUP(groupname)`
`  NETNAME(name)`
`  ACCESSMETHOD(VTAM)`
`  PROTOCOL(LU61)`
`  DATASTREAM(USER|3270|`
`  SCS|STRFIELD|`
`  LMS)`
`  RECORDFORMAT(U|VB)`
`  AUTOCONNECT(NO|YES)`
`  SECURITYNAME(name)`
`  INSERVICE(YES)`
	`DFHTCT  TYPE=SYSTEM`
	`  ,SYSIDNT=sysidnt)`
	
	`  ,NETNAME=name`
	`  ,ACCMETH=VTAM)`
	
	`  ,DATASTR=({USER|3270|`
	`  SCS|STRFIELD|`
	`  LMS})`
	`  ,RECFM={U|VB}`
	`  [,CONNECT=AUTO|ALL]`
	`  ,XSNAME=name`

Each individual session is then defined as follows:	**Each individual session is then defined as follows:**

`DEFINE`
`  SESSIONS(csdname)`
`  GROUP(groupname)`
`  SESSNAME(name)`
`  CONNECTION(sysidnt)`
`  NETNAMEQ(name)`
`  PROTOCOL(LU61)`
`  SENDCOUNT(0|1)`
`  RECEIVECOUNT(1|0)`
`  SENDSIZE(size)`
`  RECEIVESIZE(size)`
`  BUILDCHAIN(N|Y)`
`  OPERID(operator-id)`
`  OPERPRIORITY(number)`
`  OPERRSL(number)`
`  OPERSECURITY(number)`
`  IOAREALEN(value)`
`  SESSPRIORITY(number)`
	`DFHTCT  TYPE=TERMINAL`
	
	`  ,TERMIDNT=name`
	`  ,SYSIDNT=sysidnt`
	`  ,NETNAMQ=name`
	`  ,TRMTYPE=LUTYPE6`
	`  ,SESTYPE= SEND|RECEIVE`
	
	`  ,BUFFER=size`
	`  ,RUSIZE=size`
	`  ,CHNASSY={NO|YES}`
	`  ,OPERID=operator-id`
	`  ,OPERPRI=number`
	`  ,OPERRSL=number`
	`  ,OPERSEC=number`
	`  ,TIOAL=value`
	`  ,TRMPRTY=number`
	`  ,TRMSTAT=TRANSCEIVE`

Defining compatible IMS and CICS nodes

To define IMS-CICS ISC links, you should understand the relationship between the way remote systems and sessions are defined in CICS and the way they are defined in IMS.

Resource Definition Online (RDO) terms are used in the following explanation of the compatibility requirements. For the equivalent macro-level operands, see “Defining IMS-CICS LU 6.1 links” on page 613.

System names

The network name of the CICS system is specified in the APPLID operand of the DFHSIT macro.

The name can be provided as an override during CICS startup or in the APPLID operand of the DFHTCT TYPE=INITIAL macro. This name must be specified in the NAME operand of the TERMINAL macro or on the ETO logon descriptor that defines the CICS system. This name must be known to the IMS master terminal operator as it is required to be specified within those IMS master terminal operator commands (for example, **/OPNDST**) that reference sessions between IMS and CICS.

The network name of the IMS system is specified in the APPLID operand of the IMS COMM macro. You must specify this name in the NETNAME operand of the **DEFINE CONNECTION** command that defines the IMS system.

Number of sessions

For statically defined terminals in IMS, the number of parallel sessions that are required between the CICS and IMS system must be specified in the SESSION operand of the IMS TERMINAL macro.

Each session is then represented by a SUBPOOL entry in the IMS VTAMPOOL. In CICS, each of these sessions is represented by an individual session definition.

Session names

Each CICS-IMS session is uniquely identified by a *session-qualifier pair*, formed from both the CICS and IMS session names.

The CICS session name is specified in the SESSNAME operand of the **DEFINE SESSIONS** command. For sessions that are to be initiated by IMS, this name must correspond to the ID parameter of the IMS **/OPNDST** command for the session. For sessions initiated by CICS, the name is supplied on the CICS **/OPNDST** command and is saved by IMS.

The IMS session name is specified in the NAME operand of the IMS SUBPOOL macro. You must make the relationship between the session names explicit by coding this name in the NETNAMEQ operand of the corresponding **DEFINE SESSIONS** command.

Recommendation: For operational convenience, use the same CICS and IMS names for a session.

Other session parameters

The following additional operands of the **DEFINE CONNECTION** and **DEFINE SESSIONS** commands are also significant for CICS-IMS sessions.

SENDSIZE

- If CICS is the *primary* half session, specifies the maximum request unit (RU) that CICS sends to the remote IMS system. This value must be less than or equal to the value specified by the RECANY parameter in the IMS COMM macro.
- If CICS is the *secondary* half session, specifies the maximum RU that CICS receives from the remote IMS system. This value must be less than or equal to the value specified by the OUTBUF parameter in the IMS TERMINAL macro.

RECEIVESIZE

- If CICS is the *primary* half session, specifies the maximum request unit (RU) that CICS receives from the remote IMS system. This value must be less than or equal to the value specified by the OUTBUF parameter in the IMS TERMINAL macro.
- If CICS is the *secondary* half session, specifies the maximum RU that CICS sends to the remote IMS system. This value must be less than or equal to the value specified by the RECANY parameter in the IMS COMM macro.

BUILDCHAIN(N|Y)

Specifies whether multiple RU chains are to be assembled before being passed to the application program. If Y is specified, a complete chain is passed to the application program in response to each RECEIVE command, and the application must perform any required deblocking. If N is specified, a single RU is passed to the application program in response to each RECEIVE command.

Recommendation: BUILDCHAIN(Y) is recommended, even when IMS record mode (VLVB) is used, because the logical records produced as IMS output might not coincide with RU boundaries.

DATASTREAM(USER)

Specifies a data stream profile when CICS is communicating with IMS using the START command (asynchronous processing). CICS messages generated by the START command always cause IMS to interpret the data stream profile as input for component 1. This parameter is required.

The data stream profile for distributed transaction processing can be specified by the application program using the DATASTR option of the BUILD ATTACH command.

RECORDFORMAT(U|VB)

Specifies the type of chaining that CICS is to use for transmissions that are initiated by START commands (asynchronous processing) on a particular session.

Two types of data handling algorithms are supported between CICS and IMS:

- Chained

  Messages are sent as SNA chains. The user can use private blocking and deblocking algorithms. This format corresponds to RECORDFORMAT(U). When IMS is the receiver, a chain of RUs is interpreted as the complete single-segment message.

- Variable length variable blocked records (VLVB)

  Messages are sent in variable-length variable-blocked format with a halfword length field before each record. This format corresponds to RECORDFORMAT(VB). When IMS is the receiver, the input deblocked element is treated as the input segment.

The data stream format for distributed transaction processing can be specified by the application program using the RECFM option of the BUILD ATTACH command.

SENDCOUNT and RECEIVECOUNT

These operands are used to specify whether the session is a SEND session or a RECEIVE session. (In macro-level definition, this is specified in the SESTYPE=SEND|RECEIVE operand.)

A SEND session is one in which the local CICS is secondary and is the contention winner. It is specified by:

```
SENDCOUNT(1)
RECEIVECOUNT(0)
```

A RECEIVE session is one in which the local CICS is primary and is the contention loser. It is specified by:

```
SENDCOUNT(0)
RECEIVECOUNT(1)
```

Recommendation: SEND sessions are recommended for all CICS-IMS sessions.

You need not specify a SENDPFX or a RECEIVEPFX; the name of the session is taken from the SESSNAME operand.

The following table shows the relationship between the CICS Resource Definition Online and IMS definitions of an ISC link. Related operands are shown by numbers. If CICS is communicating with an XRF IMS, NETNAME(SYSIMS) should be the USERVAR or MNPS ACB associated with the XRF IMS.

Table 127. Defining compatible CICS and IMS nodes: RDO

CICS

```
DFHIST  TYPE=CSECT
       ,SYSIDNT=CICL
       ,APPLID=SYSCICS  1

DEFINE
   CONNECTION(IMSR)     3
   GROUP(group_name)
   NETNAME(SYSIMS)      2
   ACCESSMETHOD(VTAM)
   PROTOCOL(LU61)
   DATASTREAM(USER)

DEFINE
   SESSIONS(csdname)
   GROUP(group_name)
   PROTOCOL(LU61)       4
   SESSNAME(IMS1)
   CONNECTION(IMSR)     3
   NETNAMEQ(CIC1)       5
   SENDCOUNT(1)
   RECEIVECOUNT(0)
       SENDSIZE(mmm)        7
   RECEIVESIZE(nnn)     8
   IOAREALEN(nnn,16364)

DEFINE
   SESSIONS(csdname)
   GROUP(group_name)
   PROTOCOL(LU61)       4
   SESSNAME(IMS2)
   CONNECTION(IMSR)     3
   NETNAMEQ(CIC2)       9
   SENDCOUNT(1)
   RECEIVECOUNT(0)
       SENDSIZE(mmm)        7
   RECEIVESIZE(nnn)     8
   IOAREALEN(nnn,16364)
```

IMS

```
COMM     APPLID=SYSIMS    2
         RECANY=mmm + 22  7
         EDTNAME=ISCEDT

TYPE     UNITYPE=LUTYPE6  4

TERMINAL NAME=SYSCICS     1
         SESSION=2
         COMPT1=
         COMPT2=
         OUTBUF=nnn       8



VTAMPOOL

SUBPOOL NAME=CIC1         5

NAME    CICLT1 COMPT=1

NAME    CICLT1A



SUBPOOL NAME=CIC2         9

NAME    CICLT2 COMPT=2
```

The following table shows the relationship between the CICS macro-level definitions and IMS definitions of an ISC link. Related operands are shown by numbers. NETNAME=SYSIMS should be the USERVAR or MNPS ACB associated with the IMS system if you are defining a CICS to XRF IMS session.

Table 128. Defining compatible CICS and IMS nodes: macro level

CICS	IMS
DFHIST TYPE=CSECT ,SYSIDNT=CICL ,APPLID=SYSCICS 1 DFHTCT TYPE=SYSTEM ,ACCMETH=VTAM ,SYSIDNT=IMSR 3 ,NETNAME=SYSIMS 2 DFHTCT TYPE=TERMINAL ,TRMTYPE=LUTYPE6 4 ,TRMIDNT=IMS1 ,SYSIDNT=IMSR 3 ,NETNAMEQ=CIC1 5 ,SESTYPE=SEND ,BUFFER=*mmm* 7 ,RUSIZE=*nnn* 8 ,TIOAL=(*nnn*,16364) ,DATASTR=USER DFHTCT TYPE=TERMINAL ,TRMTYPE=LUTYPE6 4 ,TRMIDNT=IMS2 ,SYSIDNT=IMSR 3 ,NETNAMEQ=CIC2 9 ,SESTYPE=SEND ,BUFFER=*mmm* 7 ,RUSIZE=*nnn* 8 ,TIOAL=(*nnn*,16364) ,DATASTR=USER	COMM APPLID=SYSIMS 2 RECANY=*mmm* + 22 7 EDTNAME=ISCEDT TYPE UNITYPE=LUTYPE6 4 TERMINAL NAME=SYSCICS 1 SESSION=2 COMPT1= COMPT2= OUTBUF=*nnn* 8 VTAMPOOL SUBPOOL NAME=CIC1 5 NAME CICLT1 COMPT=1 NAME CICLT1A SUBPOOL NAME=CIC2 9 NAME CICLT2 COMPT=2

Related concepts

“LTERM users (subpools) and components” on page 463

IMS user blocks are sets of IMS logical terminals (LTERMs) defined by the SUBPOOL macro during IMS system definition or dynamically created from ETO user descriptors.

Defining multiple links to an IMS system

You can define more than one intersystem link between a CICS and an IMS system by defining two or more connections (systems), with their associated session definitions, having the same NETNAME but different SYSIDs.

Although all the system definitions resolve to the same NETNAME, and therefore to the same IMS system, using a SYSID name in CICS causes CICS to allocate a session from the link with the specified SYSIDNT.

Recommendation: Define up to three links (that is, groups of sessions) between a CICS and an IMS system, depending upon the application requirements of your installation:

- A group of sessions for CICS-initiated distributed transaction processing (synchronous processing).

  CICS applications that use the SEND/RECEIVE interface can use the SYSIDNT of this group to allocate a session to the remote system. The session is held (“busy”) until the conversation is terminated.
- A group of sessions for CICS-initiated asynchronous processing.

  CICS applications that use the START command can name the SYSIDNT of this group. CICS uses the first “nonbusy” session to ship the start request.

  IMS sends a positive response to CICS as soon as it has queued the start request, so that the session is in use for a relatively short period. Consequently, the first session in the group shows the heaviest usage, and the frequency of usage decreases towards the last session in the group.
- A group of sessions for IMS-initiated asynchronous processing.

  This group is also useful as part of the solution to a performance problem that can arise with CICS-initiated asynchronous processing. An IMS transaction that is initiated as a result of a START command shipped on a particular session uses the same session to ship its “reply” START command to CICS. For

the reasons given in (2), the CICS START command is probably shipped on the busiest session, and, because CICS is the contention winner, the replies from IMS can create a backlog while waiting for a chance to use the session.

However, facilities exist in IMS for a transaction to alter its default output session, and a switch to a session in this third group can reduce backlog problems.

The following series of tables provide an example of defining three groups of sessions on an IMS node. The tables contain examples of defining the session groups in both CICS Resource Definition Online (RDO) format and CICS macro format.

Table 129. CICS system initialization parameters

RDO definition	Macro-level definition

`DFHSIT TYPE=CSECT`
`,SYSIDNT=CICL`
`,APPLID=SYSCICS`
	`DFHSIT TYPE=CSECT`
	`,SYSIDNT=CICL`
	`,APPLID=SYSCICS`

Table 130. CICS-initiated distributed transaction processing

RDO definition	Macro-level definition

`DEFINE CONNECTION(IMSA)`
`ACCESSMETHOD(VTAM)`
`NETNAME(SYSIMS)`
 	
`DEFINE SESSIONS(csdname)`
`PROTOCOL(LU61)`
`SESSNAME(IMS1)`
`CONNECTION(IMSA)`
`NETNAMEQ(DTP1)`
 	
`DEFINE SESSIONS(csdname)`
⋮
	`DFHTCT TYPE=SYSTEM`
	`,ACCMETH=VTAM`
	`,SYSIDNT=IMSA`
	`,NETNAME=SYSIMS`
	
	`DFHTCT TYPE=TERMINAL`
	`,TRMTYPE=LUTYPE6`
	`,TRMIDNT=IMSA`
	`,SYSIDNT=IMSA`
	`,NETNAMQ=DTP1`
	
	`DFHTCT TYPE=TERMINAL`
	⋮

Table 131. CICS-initiated asynchronous processing

RDO definition	Macro-level definition

`DEFINE CONNECTION(IMSB)`
`ACCESSMETHOD(VTAM)`
`NETNAME(SYSIMS)`
 	
`DEFINE SESSIONS(csdname)`
`PROTOCOL(LU61)`
`SESSNAME(IMS1)`
`CONNECTION(IMSB)`
`NETNAMEQ(ASP1)`
 	
`DEFINE SESSIONS(csdname)`
⋮
	`DFHTCT TYPE=SYSTEM`
	`,ACCMETH=VTAM`
	`,SYSIDNT=IMSB`
	`,NETNAME=SYSIMS`
	
	`DFHTCT TYPE=TERMINAL`
	`,TRMTYPE=LUTYPE6`
	`,TRMIDNT=IMSA`
	`,SYSIDNT=IMSA`
	`,NETNAMQ=DTP1`
	
	`DFHTCT TYPE=TERMINAL`
	⋮

Table 132. IMS-initiated asynchronous processing

RDO definition	Macro-level definition

`DEFINE CONNECTION(IMSC)`
`  ACCESSMETHOD(VTAM)`
`  NETNAME(SYSIMS)`
 	
`DEFINE SESSIONS(csdname)`
`  PROTOCOL(LU61)`
`  SESSNAME(IMS1)`
`  CONNECTION(IMSC)`
`  NETNAMEQ(IST1)`
 	
`DEFINE SESSIONS(csdname)`
`  ⋮`
	`DFHTCT TYPE=SYSTEM`
	`  ,ACCMETH=VTAM`
	`  ,SYSIDNT=IMSC`
	`  ,NETNAME=SYSIMS`
	
	`DFHTCT TYPE=TERMINAL`
	`  ,TRMTYPE=LUTYPE6`
	`  ,TRMIDNT=IMS1`
	`  ,SYSIDNT=IMSC`
	`  ,NETNAMQ=IST1`
	
	`DFHTCT TYPE=TERMINAL`
	`  ⋮`

Defining CICS transactions for IMS-CICS ISC

This topic describes the unique considerations for transactions that participate in IMS-CICS ISC.

Related reading: For complete details on the definition of CICS transactions, see the *CICS Transaction Server for z/OS CICS Resource Definition Guide*.

Defining CICS backout in-doubt processing

During the period between the sending of the syncpoint request to IMS and the receipt of the positive response, CICS does not know whether the remote system has committed. This period is known as the indoubt period.

CICS processing during the indoubt period is controlled by the `DEFINE TRANSACTION IN-DOUBT` parameter (or the `DFHPCT DTB=` parameter).

DEFINE TRANSACTION IN-DOUBT (WAIT) or DTB=(YES,WAIT) must be specified in order to ensure consistency between the IMS and CICS subsystems within an ISC network.

Defining CICS transactions for asynchronous communication to IMS

A `DEFINE TRANSACTION REMOTENAME` (or `DFHPCT TYPE=REMOTE RMTNAME`) is needed for every IMS edit name or MID name that is to be referenced by a transaction created as the result of a CICS **START** command.

The 4-character CICS transaction ID used on the **START** command is converted to the name of the IMS editor or to an MFS MID name, each of which can be up to 8 characters.

Initiating and allocating a session from CICS

CICS can initiate a session in one of several ways.

- The session can be initiated explicitly by use of `AUTOCONNECT YES` on the DEFINE CONNECTION (or `CONNECT=AUTO` on the DFHTCT TYPE=TERMINAL) macro. When CICS is initiated, it attempts to establish all sessions for which AUTOCONNECT is specified. In order for session initiation to occur, IMS must be active when CICS is initiated.
- The master terminal operator can initiate a session by entering the command:

  ```
  CEMT SET TERMINAL(tttt) ACQUIRED|COLDACQ
  ```

  where *tttt* is the SESSNAME on the `DEFINE SESSIONS` or the `TRMIDNT` on the `DFHTCT TYPE=TERMINAL`.

  If `ACQUIRED` is specified, normal resynchronization with the IMS is attempted.

  If `COLDACQ` is specified, no resynchronization is performed.

The status of the CICS half session is typed over in the display area of the CEMT INQUIRE|SET TERMINAL command. If the session is initiated successfully, the status is changed from REL to ACQ. If the attempt to initiate the session is unsuccessful, an error message is written to the transient data destination CSMT.

- A session can be initiated implicitly by an application program using the ALLOCATE command. The ALLOCATE command is used on the SEND/RECEIVE interface only.

  Recommendation: Use the ALLOCATE SYSID form of this command, because it allows CICS to select an available session.

  If a session is not immediately available, control is returned to the application program in the following situations:

  – A HANDLE CONDITION for this condition is issued by the application program.
  – NOQUEUE is specified on the ALLOCATE command.

  Otherwise, the command is queued.

  The following conditions cause a session to be "not immediately available":

  – All sessions to the specified system (or the specified session) are in use.
  – The only available sessions are not bound.
  – The only available sessions are contention losers.
- A session can be initiated by CICS automatic task initiation (ATI).

The bind parameters for CICS are built from a hard-coded model. The RECEIVESIZE and SENDSIZE parameters on the DEFINE CONNECTION/SESSION (or the RUSIZE and BUFFER parameters on the DFHTCT TYPE=TERMINAL/SYSTEM) are used to determine the primary and secondary RU sizes.

Related reading: For more information on the ALLOCATE command, and subsequent processing, see *CICS Transaction Server for z/OS CICS Intercommunication Guide*.

Other ways of initiating a session

Sessions that are not initiated by CICS can be initiated either by IMS, by the VTAM operator, or Tivoli NetView for z/OS.

Terminating a session from CICS

Sessions can be terminated by CICS only by using master terminal operator commands.

The master terminal operator can use the CEMT command to release the session by entering:

```
CEMT SET TERMINAL(tttt) [RELEASED|OUTSERVICE]
```

In the example, *tttt* is the SESSNAME on the DEFINE SESSIONS or the TRMIDNT on the DFHTCT TYPE=TERMINAL.

If the master terminal operator specifies RELEASED, the session terminates when any active transaction completes and the session is in a between-brackets state.

If the master terminal operator specifies OUTSERVICE, the session also terminates when any active transactions complete and the session is in a between-brackets state. However, in this case, the session is taken out of service and cannot be used until the master terminal operator places it back into service.

If RELEASED is specified, it initiates an orderly termination between IMS and CICS. Although, from the CICS view, the session might appear to be in warm-start state, it is actually in cold-start state as a result of the SBI/BIS flow.

If OUTSERVICE is specified, the session is left in a warm-start state; that is, CICS requests resynchronization upon reinitiation. In order to initiate a session in cold-start mode, the CICS master terminal operator must specify:

```
CEMT SET TERMINAL (tttt) COLDACQ.
```

You can specify RELEASED and OUTSERVICE together.

Recommendation: Although orderly session termination can occur as a result of the CICS application program issuing EXEC CICS DISCONNECT, this approach is not recommended in a normal application. However, an operator-control type of application can be written to use this function.

Any messages relative to the session's termination are sent to transient data destination CSMT.

Related tasks

"Initiating and allocating a session from CICS" on page 620
CICS can initiate a session in one of several ways.

Related reference

"Symmetrical session shutdown for LU 6.1 (SBI and BIS)" on page 540
Two data flow control commands allow a symmetrical and orderly termination for peer level LU 6.1 half sessions: stop bracket initiation (SBI) and bracket initiation stopped (BIS).

Designing CICS applications for ISC

CICS differs from IMS in that many of the flows that must occur on an ISC session occur under the control of the CICS application program.

The CICS application must do the following:

- Build outbound ATTACH function management headers
- Supply fields for SCHEDULER function management headers (if required)
- Examine inbound ATTACH function management headers and process their contents
- Issue sync points at appropriate intervals during the application program's processing.

The SNA bracket, SEND/RECEIVE, and sync-point protocols associated with CICS-generated messages depend upon the EXEC commands (The EXEC commands are SEND/RECEIVE for synchronous and START/RETRIEVE for asynchronous.) used by the application program, and when the sync points are issued during the application flow.

Information is inserted into the ATTACH and SCHEDULER function management header DPN, RDPN, PRN, and RPRN fields for message editing and routing by IMS system code, and can be overridden only by using the IMS Message Format Service (MFS). For CICS, insertion of information into the ATTACH and SCHEDULER function management headers is performed under the control of the CICS application.

Application-related concepts

This topic describes some application-related concepts to establish a common frame of reference for both CICS and IMS users.

Subsystem design: direct-control versus queued

CICS is a direct-control subsystem, while IMS is a queued subsystem. That is, CICS accepts data entered from a terminal, and as a result, invokes the appropriate application program to process that data.

The terminal and other system resources are owned by the invoked application until the application task completes its processing. Information that results from processing is held in main storage rather than being queued. The implication for ISC is that the CICS application program is directly involved in generating the appropriate SNA data flow control, sync point, and response protocols, and in controlling most system services.

In contrast, IMS is a queued subsystem. In this case, all input and output transactions and message switches are queued by the IMS control region on behalf of the related IMS applications and terminals. Thus, the input and output of a message to or from the terminal are asynchronous from the processing of the message. (However, the processing might appear to be synchronous to the terminal because of the way in which the message is defined to IMS; for example, a response-mode message or a conversation might appear to be synchronous.) The implication for ISC is that the SNA protocols and many system services are handled under the control of IMS system code; the application program does not need to provide them.

Synchronous and asynchronous processing on ISC VTAM links

Message transmission can be synchronous or asynchronous between the terminal entering the message and the receiving subsystem. Messages can also be either synchronous or asynchronous with respect to the ISC session.

Note: This topic applies to messages that are transmitted on ISC VTAM links. For information that applies to ISC TCP/IP links, see "ISC communication with CICS over TCP/IP" on page 593.

From the point of view of the CICS application, using the CICS SEND/RECEIVE EXEC commands produces synchronous processing on the session, while the CICS START/RETRIEVE EXEC commands result in asynchronous processing. When CICS is the front-end subsystem, both processing types are supported. When IMS is the front-end subsystem, only asynchronous processing can occur, unless the special support for front-end/back-end system utilization provided by IMS Front-End Switch exit routine is used.

The following table shows how the CICS START/RETRIEVE and SEND/RECEIVE EXEC commands are supported from a CICS point of view.

Table 133. The SEND/RECEIVE and START/RETRIEVE commands

Message type	CICS to IMS	IMS to CICS
SEND/RECEIVE (synchronous)	YES	YES
START/RETRIEVE (asynchronous)	YES	YES

The following table shows the differences in the CICS application program interface when using SEND/RECEIVE or START/RETRIEVE.

Table 134. CICS API for SEND/RECEIVE and START/RECEIVE

Attribute	SEND/RECEIVE	START/RETRIEVE
Mode	Synchronous with (SEND INVITE) or without (SEND LAST) reply	Asynchronous
FMH	ATTACH	ATTACH SCHEDULER
Name length	Supports 8-byte names	Supports 4-byte names"1" on page 624
Change Direction (CD)	Supported	CICS front-end cannot send CD. IMS sends CD to CICS if COMPT=SINGLE2 or MULT2.
Multiple chains per message (IUTYPE)	Supported"2" on page 624	Not supported
IMS component selection (DATASTREAM)	Supported	Not supported for input. Input component=1. Output component=any.
RECORDFORMAT:"3" on page 625 Undefined (U)	Supported"4" on page 625	Not supported

Table 134. CICS API for SEND/RECEIVE and START/RECEIVE (continued)

Attribute	SEND/RECEIVE	START/RETRIEVE
RECORDFORMAT:"3" on page 625 RU	Supported"4" on page 625	Not supported
RECORDFORMAT:"3" on page 625 VLVB	Supported	Supported
RECORDFORMAT:"3" on page 625 CHAIN"5" on page 625	Supported	Supported
FMH built by	Application program (BUILD ATTACH)	System: DPN, PRN, RDPN, RPRN specified in EXEC START command"6" on page 625
Access to FMH	Yes (EXTRACT Attach)	Any supplied parameters available on EXEC RETRIEVE command"6" on page 625
Relation to ISC session	Directly related (ALLOCATED)	Independent
IMS transaction types supported	Response mode Conversation mode Message switches"7" on page 625 Commands Nonresponse, nonconversational"7" on page 625	Nonresponse mode Nonconversation Message switches Commands"8" on page 625
Recoverable I/O	Supported	Supported
Nonrecoverable I/O	Supported	Supported
Multisegment input	Supported	Single segment only
Multisegment output	Supported	Single segment only
IMS edits	Basic edit	Basic edit
Available	ISC Edit	ISC Edit
MFS: Without paging	Supported	Supported
MFS: Autopaged input: Single chain	Supported	Supported
MFS: Autopaged input: Single chain: Multichain	Supported	Not supported
MFS: Autopaged output	Supported	Not supported
MFS: Demand-paged output	Supported	Not recommended

Notes:

1. Names can be embedded in the message text. The DPN field can be converted from a 4-character to an 8-character name by using DEFINE TRANSACTION REMOTENAME (or DFHPCT TYPE=REMOTE).
2. Only used to send or receive MFS autopaged input or output messages.

3. The ATTACH parameters are defined in the CONNECTION definition. However, the application using START/RETRIEVE must understand and provide the correct data format within the application.
4. Marked as reserved by CICS, but can be specified on the BUILD ATTACH.
5. Chained output from IMS requires the use of MFS.
6. The RDPN and RPRN are limited to 4-character names.
7. Nonresponse, nonconversational, and message switch input use SEND LAST only. CICS acquires replies to nonresponse and nonconversational transactions using RECEIVE.
8. /DIS, /RDIS, /FOR only.

Related concepts

"Relationship of ISC and IMS execution modes" on page 461
Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

Related reference

"ATTACH FM header format" on page 557
The format of the ATTACH FM header is defined in the following table.

Front-End Switch exit routine (DFSFEBJ0) (Exit Routines)

Principal and alternate facility

The functions of CICS synchronous EXEC terminal control commands (such as SEND, RECEIVE, CONVERSE) are exercised against a principal or an alternate facility.

In CICS, the principal facility for a task is the terminal or ISC session that is made available to the application program when the task is initiated. The alternate facility is the ISC session that is allocated as needed by the application program. Commands issued against the principal facility are issued without the use of the SESSION (name) option. Commands issued against the alternate facility are issued with this option.

When CICS is the front-end subsystem, the user terminal is the principal facility. The session allocated to IMS is the alternate facility. However, when a restart transaction is attached from the ISC session, the session is the principal facility. The restart transaction does not have direct access to the user terminal.

Related concepts

"Recovery and restart concepts" on page 635
This topic describes the system and user functions that must be performed to recover an ISC session after a session, system, or application failure and assumes the reader understands the role of the STSN command in session resynchronization.

CICS versus IMS conversation mode

In IMS, conversation mode is an attribute of a transaction in which the conversation is carried on through the transfer of a scratchpad area (SPA) and an associated message between the terminal and transaction message queues.

Each transfer of information to the transaction message queue and back to the terminal queue results in a sync point, and causes the commitment of resources and the release of locked resources. Also, all of the steps of an IMS conversation must occur entirely within a bracket. EB cannot be sent at any intermediate point in the conversation, because the receipt of EB causes IMS to discard the output message, terminate the conversation, and invoke the conversational abnormal termination exit. During a conversation, if a primary resource name (PRN) parameter is supplied in the input function management header, it is ignored, because the transaction code is carried in the SPA.

In CICS, a conversation is a series of interactions within a bracket between a terminal and an application. End-bracket occurs upon termination of the application. CICS also permits a "pseudo-conversation," which is a sequence of transactions between a terminal and an application. Each transaction has one terminal input and one terminal reply within its own bracket. Either of these types of conversations can interact with IMS on the ISC session between the requests and replies by using the EXEC CONVERSE

command. A message with end-bracket being sent to the terminal by CICS also causes an end-bracket message to be returned on the ISC session to IMS, if IMS has not previously sent an end-bracket status on the reply.

The differences in the way in which conversations are defined in CICS and IMS have the following implications for ISC:

- A terminal connected to CICS can only be held synchronously with the ISC session and IMS transaction processing if the CICS transaction is conversational in the CICS sense.
- If more than one input and output is to occur on the ISC session within a single bracket, the CICS transaction can interact with an IMS transaction that is defined as conversational. However, in the event of a session failure, the conversation might not be restartable.
- If only one input and output is to occur on the ISC session within a single bracket, the CICS transaction can interact with an IMS transaction defined as response mode. A series of transactions that make up a CICS pseudo-conversation can interact on an ISC session with a series of IMS transactions defined as response mode.

IMS conversational mode requires that IMS terminate the conversation by sending the last reply with RQD2,EB. Intermediate conversational messages are sent RQE2,CD. The CICS application can decide to terminate the conversation normally by issuing SYNCPOINT or RETURN. CICS can then terminate the conversation by sending LUSTATUS RQD2,EB in response to the last conversational output from IMS. In this case, the LUSTATUS acts as the acknowledgment for the RQE2,CD sent by IMS and closes that logical unit of work. The RQD2,EB causes IMS to respond with DR2 to close the bracket, dequeue the output message, and notify the Conversational Abnormal Termination exit routine that the conversation has been terminated normally by the remote program.

Related concepts

"Recovery and restart concepts" on page 635
This topic describes the system and user functions that must be performed to recover an ISC session after a session, system, or application failure and assumes the reader understands the role of the STSN command in session resynchronization.

Sending IMS commands from CICS

IMS commands can be entered from any terminal authorized to use them.

For terminals that are defined statically, the Resource Access Control Facility (RACF) (or an equivalent product) is used for authorization. For terminals that are defined dynamically using ETO, this authorization is validated using RACF.

However, permitting a CICS application to issue IMS commands through the ISC session introduces IMS dependencies into the CICS application program. The consequences of this decision should be weighed before this facility is used.

A CICS application program can issue IMS commands by using the SEND/RECEIVE application program interface.

Exception: A CICS application program cannot issue the IMS commands **/DISPLAY**, **/RDISPLAY**, **/FORMAT**, and **/TEST**.

The IMS command verb is embedded in the message. It cannot be specified in the PROCESS parameter of the BUILD ATTACH command.

/DISPLAY, **/RDISPLAY**, and **/FORMAT** can be sent using START/RETRIEVE or SEND/RECEIVE. These commands differ from other IMS commands in that replies to these commands are queued by IMS rather than being immediately processed.

If these commands are issued with SEND (without LAST), IMS returns an LUSTATUS NO-OP as the reply to force end-bracket. IMS replies to these commands by sending the reply ATTACH BB/EB. CICS must use RECEIVE to obtain the reply. If a command is issued with SEND LAST, the reply message, when it is returned, is processed by a CICS transaction whose name has been previously specified in the RPROCESS field of the BUILD ATTACH.

If a command is issued using START/RETRIEVE, the reply is returned with both the ATTACH and SCHEDULER FM headers.

You can use the **/TEST** command for testing communication protocols and editing facilities on an ISC session when IMS is a back end to a CICS front end. Test mode requires the SEND/RECEIVE application program interface to be used.

Related concepts

"Issuing IMS commands from an ISC session" on page 460
Although available to the ISC session, IMS operator commands are primarily intended for use by appropriately authorized operators of IMS master terminals and remote terminals that are directly attached to an IMS system.

"Overview of the Extended Terminal Option" on page 61
The Extended Terminal Option (ETO) of IMS allows you to add VTAM and ISC TCP/IP terminals and users to your IMS without predefining them during system definition.

IMS security (System Administration)

Related tasks

"Using IMS test mode for ISC VTAM sessions" on page 460
You can test ISC VTAM communication protocols and editing facilities by putting a back-end IMS system into test mode.

Sync points

ISC session protocols define RQ*2 requests and their associated responses (DR2 or exception DR2) as sync point requests and responses.

These requests and responses are functionally independent from RQ*1 requests and their associated responses (DR1 and exception DR1), because these latter requests and responses do not cause a sync point. The sync-point responses (DR2) are used between ISC session partners to ensure that both partners' subsystem sync point managers can commit or back out recoverable resources in synchronism.

All messages sent or received on an ISC session are defined as either recoverable or nonrecoverable, depending on the message type. The session response protocols are used to ensure that both partners of an ISC session mutually understand and agree with the recoverability attributes associated with each message. The response protocols used must be consistent with the message type and are enforced by the sending and receiving subsystem sync point managers.

When one session partner commits a message, the other session partner is notified by positive session sync-point responses. When a session partner backs out a message as the result of errors detected during input or output processing, the other session partner is notified either by session termination, by an exception response sent by the receiving subsystem, or by an LUSTATUS-function abort sense code sent by the sending subsystem. Backout results in discarding the currently active message and resetting of the associated ISC data flow control (DFC) and ATTACH states to those of the last sync point.

Because CICS is a direct-control subsystem, synchronous transactions directly control the ISC session. Therefore, backout during synchronous transaction processing also results in backing out application updates made since the last sync point.

Queued subsystems and asynchronous application output do not provide direct control of the ISC session to the transaction; therefore commit or backout does not affect application updates that are made asynchronously to the message on the ISC session. Because IMS is a queued subsystem, session sync points are separate from application sync points, and always occur outside control of the executing application.

Sync points on IMS input

For any type of input, IMS does not schedule the intended transaction until the complete input message is successfully received. Processing errors and session failure prior to the receipt of the complete input message cause the entire message to be discarded or backed out and have no effect on other

recoverable IMS resources, such as databases. However, the input message cannot be canceled by ISC session failures or protocols after the complete message is received, enqueued, and made available for scheduling. Only the current input message is backed out, even when several consecutive input (nonrecoverable) messages are received and enqueued because of a previous input sync point was requested. The definition and relationship of ISC input sync points to the application sync points depend upon whether the internal IMS execution mode for the input is synchronous or asynchronous.

Synchronous transactions from CICS are sent using SEND/RECEIVE, which generates ATTACH with CD. The following IMS transaction types are synchronous:

- Response mode transaction
- Conversational mode transaction
- IMS commands
- Test mode input

Asynchronous transactions from CICS are sent using SEND LAST, which generates ATTACH with EB, or START/RETRIEVE, which generates ATTACH SCHEDULER with EB. The following transaction types are asynchronous:

- Nonresponse mode transaction
- Nonconversational mode transaction
- IMS message switch

For IMS, the sync-point response returned to CICS can indicate successful receipt of the message or can indicate the results of IMS processing. For asynchronous input to IMS, the sync-point response returned to CICS indicates only that the input message has been successfully enqueued and that the responsibility for message recovery is assumed by IMS. For synchronous input to IMS, the sync-point response returned to CICS indicates that the transaction has executed successfully and a sync point has been taken, even though transaction execution is independent of the session. IMS reflects this by returning the input sync-point response only after the application-inserted reply message is made available for output as the result of a successful application sync point.

Sync points on IMS output

For output, IMS commits the output message when the requested sync-point response is returned by CICS. This means that the message has been successfully sent and dequeued and the session sync point information has been updated as appropriate. IMS backs out (depending upon the sense code used) the output message when an exception response is returned by CICS. In this case, the message is returned to the message queue for retransmission, and the associated DFC and Attach states are reset to those of the last ISC sync point. Backout on the ISC session does not affect the IMS application program or other recoverable resources, such as database updates. Backout of IMS conversational-mode output, in most cases, causes termination of the IMS conversation and the Conversational Abnormal Termination exit routine to be invoked. Based on the contents of the conversational SPA, user-provided exit logic might schedule another IMS transaction to back out database changes.

CICS sync points

For CICS, sync points occur under the control of the application program and can be issued at any time. For synchronous processing, when a sync-point request is issued by the application program, CICS logs the completion of the logical unit of work (that is, the resources are committed). In addition, when the SYNCPOINT command is issued within a CICS application, CICS is responsible for appending the ISC message that was created as a result of previous processing to the SNA response request (RQD1 or RQD2) that IMS expects. However, when a CICS asynchronous transaction is scheduled as a result of IMS input, the mirror transaction has already issued a sync point, that causes the appropriate response to be returned to IMS. Any sync points issued subsequently by the application have no effect on the ISC session.

Related concepts

“Keeping half sessions synchronized” on page 506

Sync-point responses (DR2) are used between ISC session partners to ensure that both partners' sync-point managers can commit or back out recoverable resources synchronously.

“Relationship of ISC and IMS execution modes” on page 461
Because the terms "synchronous" and "asynchronous" have slightly different connotations within IMS and ISC, the following topics explain the relationship of these execution modes.

“Logical unit of work” on page 636
A logical unit of work is any processing that occurs between sync points.

Coding function management headers for CICS

CICS uses some of the same SNA-defined function management header fields that are used by IMS.

Unlike IMS, the CICS application program must prepare these function management headers and send them to IMS. CICS, itself, does not send them.

Both ends of an ISC session have an "attach manager" that performs the functions requested in the FMH.

Related reference

“Function management headers” on page 541
In SNA, function management (FM) headers are an optional part of the request unit sent over a link. This topic describes the FM headers supported by IMS on ISC sessions.

ATTACH function management header

This topic describes the ATTACH function management header fields. The primary fields that CICS uses are ATTDPN, ATTPRN, ATTRDPN, and ATTRPRN.

ATTDPN

The CICS transaction specifies the outbound destination process name field (ATTDPN) in the PROCESS field of the BUILD ATTACH command.

This field, when sent by CICS to IMS, contains the name of an IMS editor (basic edit or ISC edit) or an MFS MID name that is to receive the inbound message. When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return destination process name field (ATTRDPN) on the originating outbound message. If this is a reply to a nonresponse or nonconversational message sent to IMS with SEND LAST, this field identifies the transaction to be initiated to process this reply. In other cases, this value can be a CICS transaction code that identifies a restart transaction. If this value is not available, CICS uses the first four characters of the incoming data stream as the transaction code. If the reply is returned on the same session as that of the incoming message, IMS system code automatically wraps the incoming RDPN field into the DPN field of the reply message.

CICS ignores the ATTDPN if it is received in a reply returned from IMS.

Exception: After a session restart or a reply to a message sent with SEND LAST, ATTDPN is used to attach a CICS transaction.

When used with the SCHEDULER, the ATTDPN contains the DPN of the SCHEDULER model. In this case, the remaining ATTACH parameters are not sent by IMS, but rather, the information is supplied on the associated SCHEDULER model parameters.

Related reference

“SCHEDULER function management header” on page 632

The SCHEDULER function management header can be used to send asynchronous messages between IMS and CICS.

ATTPRN

The CICS transaction specifies the primary resource name field (ATTPRN) in the RESOURCE field of the BUILD ATTACH command.

This field, when sent by CICS to IMS, names an IMS LTERM or transaction code that is the message destination. When this parameter is omitted, IMS uses the first eight characters of the data stream to identify the message destination.

Recommendation: To maintain consistency between IMS and CICS, use this field to name an LTERM for IMS message switches and place transaction codes in the first eight characters of the data stream.

When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return primary resource name (ATTRPRN) on the outbound message. CICS ignores the ATTPRN if it is received in a reply returned from IMS.

Exceptions:

- After session restart, ATTPRN is used to determine the identity of the terminal that originated this transaction and must be acquired during the restart.
- When the reply is to a nonresponse or nonconversational transaction issued by CICS with SEND LAST, ATTPRN is used.

If the reply is to be returned on the same session as that of the incoming message, IMS automatically wraps the value received on the incoming RPRN field into the PRN field. MFS can be used to set or override this value on IMS output.

ATTRDPN

The CICS transaction specifies the return destination process name (ATTRDPN) in the RPROCESS field of the BUILD ATTACH command.

When CICS sends this field to IMS, it contains the code of a transaction to be executed in the event that a session restart is required or a reply is to be returned to a message that CICS sent to IMS using SEND LAST. IMS sends this to CICS when a next MID is specified within an MFS message output descriptor (MOD). When CICS receives the ATTRDPN field on an input message, it can be examined by the EXTRACT ATTACH command. This value should be saved and returned to IMS as the DPN on subsequent replies.

ATTRPRN

The CICS transaction can place the identification of the originating terminal in the ATTRPRN field.

This identification is wrapped into the PRN field by IMS if a session restart is required or a reply is required to a message sent to IMS by CICS with SEND LAST. The CICS transaction places this value in the field by using the RRESOURCE field of the BUILD ATTACH and examines this field by using the EXTRACT ATTACH command.

For asynchronous unsolicited output, IMS uses the source LTERM name as the default, if a reply is necessary. MFS can be used to set or override this value on any type of output. When a CICS application receives this value, the value should be saved and returned as the PRN field for subsequent replies.

ATTDQN and ATTDP

When sent to CICS from IMS, on the ATTACH for a demand-paged output message, ATTDQN contains a unique message identifier.

This field is only used for IMS demand paging. The CICS transaction specifies the destination message ID (ATTDQN) for subsequent demand-paged requests as the QNAME value in the returned QMODEL FM header. When CICS receives ATTDQN from IMS, the ATTDP field is set to 1. These fields are not set by IMS on any other conditions nor are they accepted by IMS on input.

ATTIU

The CICS transaction specifies the interchange unit code (ATTIU) in the IUTYPE field of the BUILD ATTACH command.

Two values can be specified: single chain, which applies to all non-MFS input or output, and multichain, which applies only to MFS-autopaged input and output. Single chain should be used unless the input to IMS is multiple-page MFS-autopaged input.

ATTDSP

The CICS transaction specifies the data stream profile (ATTDSP) in the DATASTR field of the BUILD ATTACH command to determine the IMS component.

This field can contain the following values:

X'00'
Identifies IMS component 1

X'01'
Identifies IMS component 2

X'02'
Identifies IMS component 3

X'03'
Identifies IMS component 4

When received by IMS, this input component determines the default output component, the input security requirements, and whether MFS can be used for input. The output component determines whether MFS is used for output, and the protocols that are to be sent on asynchronous output are sent ATTACH SCHEDULER. On output, IMS sets this value to the value of the output LTERM.

ATTDBA

The CICS transaction specifies the deblocking algorithm (ATTDBA) in the RECFM field of the BUILD ATTACH command.

ATTACC

The ATTACC parameter is not supported on an ISC session and causes session termination if sent.

The following table summarizes the source of the values placed in the ATTACH FM header fields.

Table 135. Source of values placed in the ATTACH FM header fields

Major field	**A**"1" on page 632	**B**"2" on page 632	**C**"3" on page 632
ATTDPN"4" on page 632	PROCESS	DPN	
ATTPRN	RESOURCE	PRN	
ATTRDPN	RPROCESS	RDPN	
ATTRPRN	RRESOURCE	RPRN	
ATTDQN	QUEUE	Message ID"5" on page 632	
ATTIU	IUTUPE"6" on page 632	Paging"6" on page 632	
ATTDSP	DATASTR"7" on page 632		COMPTn"8" on page 632
ATTDBA	RECFM"7" on page 632		VLVB
			DPM-Bn"9" on page 632

Notes:

1. CICS EXEC BUILD/EXTRACT ATTACH.
2. Set by IMS MFS; or, if MFS is not used, and the output is asynchronous unsolicited output, the RPRN defaults to the source LTERM name.
3. IMS COMPTn in TERMINAL/NAME macros.
4. For asynchronous processing, this field indicates "SCHEDULER follows" by containing the value X'02'. In this case, the remaining header fields are not contained in the ATTACH FM header, but are contained instead in the concatenated SCHEDULER FM header.
5. Message ID for MFS demand paging.
6. Value is set depending on MFS autopage.
7. Second byte of these halfword binary fields is used.
8. Components numbered 1 through 4.
9. DPM-Bn permits MFS to be used. Using MFS MODE=STREAM on output changes the ATTDBA to chain mode; otherwise, IMS uses the default of VLVB.

SCHEDULER function management header

The SCHEDULER function management header can be used to send asynchronous messages between IMS and CICS.

On input, receipt of the SCHEDULER FMH causes CICS to invoke the mirror transaction to schedule the asynchronous transaction that is to process the input. On output, it is generated as a result of CICS using the START command with its associated fields as follows:

SCDDPN

The CICS transaction specifies the value to be placed in the outbound destination process name field (SCDDPN) in the TRANSID field of the START command.

This field, when sent by CICS to IMS, contains the name of an IMS editor (MFS, basic edit, or ISC edit) that is to receive the inbound message. If SYSID is not coded on the START command, this TRANSID can be modified (to contain an 8-character name, for example) as a result of the RMTNAME option in the program control table (PCT).

When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return destination process name field (SCDRDPN) on the outbound message. IMS automatically wraps the incoming RDPN field into the DPN field of the reply message. Alternatively, this field might have been set by the IMS MFS. This value is a CICS transaction code. If this field is not supplied on input to CICS, CICS uses the first four characters of the input data stream as the transaction code.

SCDPRN

The CICS transaction specifies the primary resource name field (SCDPRN) in the TERMID field of the START command.

This field, when sent by CICS to IMS, names an IMS LTERM or transaction code that is the message destination. If this field is omitted when CICS sends a message to IMS, IMS examines the first eight bytes of the incoming message to extract the message destination.

Recommendation: To maintain consistency between IMS and CICS, use this field to name an LTERM for IMS message switches and place transaction codes in the first eight characters of the data stream.

When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return primary resource name (SCDRPRN) on the outbound message or a value placed into the field by MFS. This field contains a value that identifies the terminal that is to be used as a principal facility and to which the reply should be sent.

SCDRDPN

The CICS transaction specifies the return destination process name (SCDRDPN) in the RTRANSID field of the START command.

When CICS sends this field to IMS, it contains the name of the asynchronous transaction to be scheduled by the CICS mirror transaction when the reply is received.

For unsolicited asynchronous output, the RDPN can be set by MFS to indicate the next message input descriptor (MID) to be used for subsequent replies. If this parameter is received as input by CICS, this parameter should be saved and returned to IMS in the TRANSID (DPN) of the START command.

SCDRPRN

The CICS transaction specifies the return primary resource name (SCDRPRN) in the RTERMID field of the START command.

When CICS sends SCDRPRN to IMS, it contains the identification of the terminal to which the return reply should be routed. On output from IMS to CICS, IMS sets SCDRPRN to the name of the source LTERM. MFS can also set SCDRPRN. The SCDRPRN value should be saved by CICS and returned to IMS as the TERMID (PRN) on any subsequent replies.

SCDDQN and SCDDP

Because asynchronous demand paging is not recommended, the SCDDQN and SCDDP fields are not used in a CICS-IMS ISC session.

The SCHEDULER header is preceded by an ATTACH header. Some mandatory ATTACH FM header fields also apply to asynchronous messages. These fields are not carried on the SCHEDULER header, but are retained on the ATTACH header. These fields are the ATTIU, ATTDSP, and ATTDBA fields.

Their values in the asynchronous environment are as follows:

- ATTIU

  CICS only supports single-chain messages between itself and IMS.
- ATTDSP

  When sent using the START command, the ATTDSP value is determined by the terminal-control table generation, and should be X'00' (component 1). For input and output, the definition of the ATTDSP field for IMS is the same as that described in “ATTACH function management header” on page 629.
- ATTDBA

  The CICS transaction specifies the deblocking algorithm (ATTDBA) in the RECFM field of the BUILD ATTACH command.

The following table summarizes the source of the values placed in the SCHEDULER FM header fields.

Table 136. Source of values placed in the SCHEDULER FM header fields

Major field	A“1” on page 634	B“2” on page 634	C“3” on page 634	D“4” on page 634
SCDDPN	TRANSID“5” on page 634		DPN	
SCDPRN	TERMID“5” on page 634		PRN	
SCDRDPN	RTRANSID“5” on page 634		RDPN	
SCDRPR	RTERMID“5” on page 634		RPRN	

Table 136. Source of values placed in the SCHEDULER FM header fields (continued)

Major field	A"1" on page 634	B"2" on page 634	C"3" on page 634	D"4" on page 634
SCDDQN	QUEUE		Note "6" on page 634	
ATTIU"7" on page 634			Note "6" on page 634	
ATTDSP"7" on page 634		DATASTR"8" on page 634		COMPTn"8" on page 634
ATTDBA"7" on page 634		RECFM		VLVB
				DPM-Bn"9" on page 634

Notes:

1. CICS EXEC START/RETRIEVE.
2. CICS DEFINE CONNECTION or DFHTCT TYPE=SYSTEM.
3. Set by IMS MFS; or, if MFS is not used, for output that is unsolicited asynchronous output, the RPRN defaults to the source LTERM name.
4. IMS COMPTn in TERMINAL/NAME macros or on an ETO logon descriptor.
5. Four-byte name; MFS supports an eight-byte name.
6. MFS paging is not recommended on an asynchronous session between IMS and CICS.
7. ATTACH FMH fields preceding SCHEDULER FMH.
8. DATASTR must be specified as USER. This causes X'00', specifying IMS component 1, to be indicated as the input component. The IMS output component can be components 1 through 4 (X'00'- X'03').
9. DPM-Bn permits MFS to be used. Using MFS MODE=STREAM on output changes the ATTDBA to chain mode; otherwise, IMS uses the default of VLVB.

Related reference

"ATTACH function management header" on page 629
This topic describes the ATTACH function management header fields. The primary fields that CICS uses are ATTDPN, ATTPRN, ATTRDPN, and ATTRPRN.

Queue model function management headers

CICS uses the QMODEL headers to access IMS MFS demand-paged messages.

QMODEL headers are not sent by IMS to CICS. For SEND/RECEIVE, the CICS application program must prepare the QMODEL function management headers in order to receive demand-paged output from IMS. Temporary storage function shipping can be used with START/RETRIEVE. No EXEC commands exist to support the coding of QMODEL FM headers. The following QMODEL headers are used for MFS demand paging:

QGET
Requests a page directly

QGETN
Requests the next sequential page

QPURGE
Requests termination of demand paging

QXFR
Sent with the requested page

QSTAT
Sent when QPURGE is received, or when the requested page number is invalid

Data descriptor function management header

IMS uses the data descriptor function management header to specify a device page (DPAGE).

This header is not used by CICS, but can be passed to a CICS transaction. A CICS synchronous transaction can build and send data descriptor FM headers to IMS as appropriate. However, no EXEC commands exist to support the coding of data descriptor FM headers.

System message process (SYSMSG) function management header

IMS uses the system message process (SYSMSG) function management header to send system messages.

System messages are appended to an ATTACH carrying DPN=SYSMSG. CICS routes these messages to a synchronous application when the session is allocated to that transaction or to destination CSMT at any other time. If routed to a transaction, that transaction is responsible for processing the SYSMSG function management header. CICS does not send SYSMSG, but a synchronous application program can cause a SYSMSG to be sent by setting the appropriate process name in the BUILD ATTACH. The application program is responsible for building the appropriate SYSMSG FMH to be appended to the ATTACH.

SYSMSG is sent by IMS for broadcast messages or for the case in which a response to input has been sent, but has had a subsequent processing error. After a response to asynchronous input, SYSMSG is sent in lieu of the reply message. During synchronous processing, SYSMSG is sent:

- After a session restart, if the transaction in progress abends during the session outage
- During MFS output, if MFS output format blocks are not available or are not valid

Error recovery procedure function management header

CICS sends and receives the error recovery procedure function management header to transmit error information from one process to another.

The header is sent after an exception response (X'0846') and carries sense codes, which are used to inform the half-session partner of the nature of the error. It is followed by an error message. CICS writes the received IMS error message to the transient data destination CSMT; IMS writes the received CICS error message to the master terminal or the input source node.

Recovery and restart concepts

This topic describes the system and user functions that must be performed to recover an ISC session after a session, system, or application failure and assumes the reader understands the role of the STSN command in session resynchronization.

The concepts of sync point, commit, backout, and logical unit of work are common to IMS and CICS. However, their meanings differ slightly within IMS and CICS.

Related concepts

“Determining session synchronism using STSN” on page 496
The need to resynchronize can be communicated during bind negotiation; two flags in the BIND request are used to determine the requirement to resynchronize and to return the half sessions to the state that existed at the time of session termination.

“Sync points” on page 627

ISC session protocols define RQ*2 requests and their associated responses (DR2 or exception DR2) as sync point requests and responses.

Logical unit of work

A logical unit of work is any processing that occurs between sync points.

Within subsystems where the application directly controls the session, such as CICS, the application sync point and the ISC session sync point occur simultaneously as the result of a single explicit or implicit application command. Within queued subsystems, such as IMS, the application processing and sync points are independent of those of the ISC session. IMS must map the application sync point to the ISC session sync-point request.

In the synchronous case (SEND/RECEIVE), CICS is the front-end subsystem. One logical unit of work includes all of the processing performed in both the IMS and CICS subsystems, from the point at which the CICS application last issued a sync point to the point at which the next sync point is issued, and a sync-point response is returned.

If the transaction is recoverable, only one message can be sent to IMS before the sync point must be requested by the CICS application. However, a series of nonrecoverable requests and replies can occur between a CICS sync-point request and an IMS response.

For recoverable messages, IMS returns this response only after the IMS application has inserted a response or conversational mode reply message and caused an application sync point. IMS then begins the next logical unit of work by sending the reply message requesting a sync point. The reply message satisfies the RECEIVE issued by the CICS transaction. The CICS transaction should complete all necessary processing prior to issuing a SYNCPOINT or RETURN (implicit sync point) command.

The application sync point causes CICS to return a sync-point response to IMS. IMS then completes the logical unit of work by dequeuing the reply message from the output message queue.

In the asynchronous case, where CICS is the front-end subsystem, for recoverable input to IMS, the input request and the returned reply message are separate logical units of work. When the CICS application issues a SEND LAST or START and requests a sync point, IMS returns the sync-point response as soon as the transaction is enqueued (made available for scheduling). IMS sends any recoverable asynchronous reply messages as they are made available by the transaction and requests a sync point on each. When the reply message is read by the CICS mirror transaction, an appropriate response is automatically returned to IMS. If the reply message is read using RECEIVE as a result of a previous SEND LAST, the sync point response is not returned to IMS until a subsequent SYNCPOINT, FREE, or RETURN command is issued.

One or more nonrecoverable transactions can be sent to IMS between CICS sync points using multiple START commands, each followed by a RETRIEVE. When the reply message is read by the CICS mirror transaction, an appropriate response is automatically returned to IMS.

The concept of logical unit of work (or work unit) is important in that session resynchronization must be performed when initiating a session and up to one work unit is considered to be indoubt on a flow from either the primary to the secondary half session or from the secondary to the primary. An indoubt work unit is one that is waiting to be committed or backed out based on the results of the session BIND and STSN flow.

Recovering outstanding message traffic after a failure

A CICS-IMS session can fail or terminate in any of several situations.

The situations in which a CICS-IMS session can fail or terminate include:

- A communication component (for example, VTAM or NCP) fails.
- The CICS or IMS subsystem fails.
- A direct VTAM or subsystem command (for example, an IMS /CLSDST or /STOP command) is entered.
- The CICS transaction or IMS transaction fails (indirectly as a result of subsequent error processing).

In each of these situations, the session failure appears similar to the remaining operative subsystems. However, the resulting recovery and resynchronization processes you must follow differ. This topic describes what happens in the event of a communication component failure, an IMS or CICS subsystem failure, or the issuing of a direct command against the session between the two subsystems.

Related concepts

"Handling transaction abends" on page 639
In addition to the considerations for session and subsystem failure, the design of CICS recovery transactions must also take into account the actions to be taken in the case of transaction termination as described in the topics that follow.

Reestablishing the session

The failed session can be restarted by either half-session partner.

When it is restarted, the relationship of the half sessions is the same as when the session failed; that is, the former primary half session continues to be primary and the former secondary half session to be secondary. Both IMS and CICS remember this orientation by logging the session qualifier pair (SQP) used to initiate the session with an indicator of the session polarity.

The CICS master terminal operator can reinitiate the session using the command:

```
CEMT SET TERMINAL (tttt) ACQUIRED
```

where *tttt* is the TERMIDNT of the CICS session as defined in the terminal control table. CICS then brings up the session, maintaining the session polarity that has been fixed by CICS system definition parameters. If the session is being initiated by an authorized IMS terminal operator's issuing the /OPNDST command, that command must specify the same subpool name (IMS local session qualifier name) and session ID (CICS local session qualifier name) that was allocated to the session at the point of failure. These are available to the operator as necessary using an IMS /DISPLAY command. IMS automatically reestablishes the session, while maintaining the same session polarity as was in effect at session failure.

If the session is being restarted with IMS in cold-start mode, and if CICS is defined to be the primary half session, CICS must initiate the restart.

When the session is reestablished, IMS attempts to reset the data flow control bracket state to its status at the last sync point before session failure. CICS allows the session to be reestablished as between-brackets or as in-brackets with IMS in SEND state.

Resynchronizing the session

When a session is active, both IMS and CICS maintain a set of SNA message sequence numbers for that session. When a failure occurs and restart is attempted, the total of sync points on the session is checked between the subsystems by the STSN command.

The CICS/IMS session resumes if both of the following are true:

- The session is being cold-started.
- The half sessions' sequence numbers agree, or are within acceptable limits.

If a session fails, a CICS transaction having pending activity for that session also fails. CICS uses dynamic transaction backout (DTB) to ensure integrity of recoverable resources. The DTB indoubt parameter must be specified as IN-DOUBT(WAIT) or DTB=(YES,WAIT) in order to ensure consistency between the IMS and CICS subsystems within an ISC network.

WAIT holds locks on recoverable resources until resynchronization occurs.

If this situation is undesirable, you can specify IN-DOUBT(BACKOUT) or DTB=YES on an IMS-CICS ISC transaction. However, doing so might cause duplicate messages to be sent to IMS after session resynchronization, and can require logic in the IMS user transaction to resolve them.

If the DTB indoubt parameter does not specify WAIT, you must specify FORCSESS on the IMS TERMINAL macro system definition OPTIONS parameter or use an authorized IMS terminal operator /CHANGE command.

Related concepts

"Determining session synchronism using STSN" on page 496
The need to resynchronize can be communicated during bind negotiation; two flags in the BIND request are used to determine the requirement to resynchronize and to return the half sessions to the state that existed at the time of session termination.

"Set-and-Test-Sequence-Numbers (STSN)" on page 926
Message resynchronization is initiated by IMS when it sends the set-and-test-sequence-numbers (STSN) command to the workstation.

Related tasks

"Defining CICS backout in-doubt processing" on page 620
During the period between the sending of the syncpoint request to IMS and the receipt of the positive response, CICS does not know whether the remote system has committed. This period is known as the indoubt period.

Processing outstanding traffic

In IMS, transaction processing is independent of session status; that is, if a message is received and successfully queued, it is always processed.

However, the way in which IMS handles subsequent replies and continues processing conversations depends on the type of message, the type of error that occurs, and whether the IMS subsystem fails.

In general, when only the session fails (and IMS does not fail), and no synchronous processing is occurring between IMS and CICS, the following is true:

- The session is bound between-brackets when it is reinitiated.
- IMS always sends or resends asynchronous replies or unsolicited asynchronous output after successful session restart.
- IMS sends or resends queued asynchronous replies resulting from /DISPLAY, /RDISPLAY, and /FORMAT commands.
- If the IMS asynchronous transaction abends during the session outage, IMS sends an error message using ATTACH SYSMSG, because exception responses are no longer possible.

For replies processed synchronously by IMS:

- If response mode or conversational output is pending, the session is bound with IMS in-brackets/SEND to permit the pending reply to be sent or re-sent.
- When in conversation mode and input is required, IMS attempts to bind in-brackets/RECEIVE. CICS negotiates this bind to a between-brackets state, causing IMS to terminate the conversation, discard the output message, and invoke the Conversational Abnormal Termination exit routine. This exit routine can invoke user processing to schedule an IMS transaction to back out any database changes resulting from previous conversational processing based on the contents of the SPA.

  When a session is reestablished and sent to between-brackets as described in the preceding paragraph, if the IMS transaction has not yet completed processing the conversational or response mode input (that is, output is not yet available), IMS terminates the session with a message to the master terminal operator requesting that the session be reinitiated (when output is available). This occurs because IMS cannot terminate the in-process transaction as required by the between-brackets bind.
- Except as mentioned, IMS cancels any commands received at the point of failure and discards pending synchronous reply messages that result from an IMS command.
- If the IMS synchronous transaction abends during the session outage, IMS sends an error message using ATTACH SYSMSG, because exception responses are no longer possible.

IMS handles these transactions in the following manner:

- After an emergency restart, IMS discards a pending reply message resulting from a nonrecoverable transaction.

- After IMS receives and enqueues an input transaction, session protocols or failures cannot cancel that transaction.
- After a reply message to a CICS synchronous transaction (response mode or conversation) is enqueued, it indicates that the results of all previous IMS processing have been committed. These updates are not backed out even if the reply message is discarded.

If a CICS transaction must be re-sent to IMS as determined by STSN processing, the CICS application, rather than the CICS subsystem itself, must make this determination and either reconstruct the transaction or request that it be reentered.

After a session fails, CICS cannot reestablish the environment in which the original transaction was executing. That is, the connection to the terminal that originally entered the transaction no longer exists. Further, the transaction has abnormally terminated; therefore, it is necessary for the CICS application programmer to define a "restart transaction" to be invoked by CICS to handle any IMS output that CICS might receive on the restarted session. When this transaction is invoked, the session, rather than the terminal, is now the primary facility. This transaction must in turn invoke an asynchronous transaction to acquire the terminal if output exists that needs to be sent to it.

The transaction code of the restart transaction to be invoked is carried in the DPN field of the function management header that is sent to CICS with the input message. This transaction code is the one specified in the RPROCESS field of the BUILD ATTACH command sent by CICS with the outbound message. This becomes the RDPN parameter automatically wrapped by IMS to the DPN field, which IMS sends on the outbound FMH unless modified by the IMS MFS.

The ID of the terminal to which CICS is to send any output reply is found in the PRN field of the incoming function management header. This is the value specified in the RRESOURCE field of the BUILD ATTACH sent by CICS with the outbound message. This becomes the RPRN parameter automatically wrapped by IMS to the PRN field of the outbound FMH unless modified by MFS. This value is used as the TERMID of the asynchronous transaction scheduled (START) by the restart transaction to acquire the terminal.

The restart transaction issues a RETRIEVE carrying both TRANSID and TERMID fields. When the transaction specified by the TRANSID is initiated, it owns the terminal specified by the TERMID as its principal facility.

Related concepts

“Handling transaction abends” on page 639
In addition to the considerations for session and subsystem failure, the design of CICS recovery transactions must also take into account the actions to be taken in the case of transaction termination as described in the topics that follow.

“Coding CICS applications for restart” on page 640
When a failure occurs on an ISC session as a result of session or subsystem failure, session restart and resynchronization can be attempted.

Handling transaction abends

In addition to the considerations for session and subsystem failure, the design of CICS recovery transactions must also take into account the actions to be taken in the case of transaction termination as described in the topics that follow.

IMS transaction abend

When an IMS transaction terminates abnormally, all changes to IMS resources that were performed by this transaction are backed out by IMS. If the transaction is synchronous (CICS SEND/SYNCPOINT), IMS sends a negative response (exception DR2) to CICS. This causes the CICS transaction to also terminate abnormally. As a result of the negative response and subsequent CICS transaction abend, CICS backs out any changes made to recoverable resources after the previous CICS sync point. An error message is also sent to CICS; CICS routes this message to the master terminal destination CSMT. The exception occurs when MFS output formats do not exist in the format library or incur an I/O error. In this case, IMS has already sent a positive sync-point response and must indicate the MFS error to CICS by using an ATTACH SYSMSG.

If the transaction is asynchronous (CICS START or SEND LAST), IMS sends an error message to CICS; CICS routes this message to its master terminal destination. The error message is preceded by either a SYSMSG or an ERP header, depending upon the type of error that occurs. If the error occurs before IMS sends a sync-point response to input, an exception response and ERP are used; if after the sync-point response, ATTACH SYSMSG is used.

CICS transaction abend

If a transaction is initiated by a CICS subsystem acting as a front end for IMS, and that CICS transaction terminates abnormally before reaching sync point, any processing performed is handled in accordance with the specification on the DEFINE TRANSACTION IN-DOUBT parameter (or DFHPCT DTB= parameter).

If an asynchronous transaction is initiated by an IMS front end and the CICS transaction abends upon receipt of the transaction, IMS is not notified. This is because the receipt of the message causes the CICS mirror transaction to return an immediate DR2 to the asynchronous input and to schedule a transaction to process it. If this scheduled transaction fails, the ISC link to IMS is no longer active and no notification is possible. CICS does, however, notify the CICS destination CSMT of the transaction failure.

During synchronous processing, if CICS is the secondary half session, the ALLOCATE command has been successfully issued, and no message is available to be sent to IMS (due, for example, to DTB causing backout), CICS automatically deallocates the session by sending LUSTATUS BB/EB to IMS.

If the synchronous transaction terminates abnormally any time after it has sent a message to IMS using the SEND/SYNCPOINT command, and that message has been enqueued on an IMS input queue, IMS processes it. When IMS attempts to send an output message to CICS, that message receives a negative response from CICS. As a result, the message can remain on the IMS output queue until it can be re-sent or until it is dequeued by the IMS master terminal operator.

Related reference

“Coding function management headers for CICS” on page 629
CICS uses some of the same SNA-defined function management header fields that are used by IMS.

“Error recovery procedure function management header” on page 635
CICS sends and receives the error recovery procedure function management header to transmit error information from one process to another.

Coding CICS applications for restart

When a failure occurs on an ISC session as a result of session or subsystem failure, session restart and resynchronization can be attempted.

If session resynchronization (STSN processing) is unsuccessful and the session cannot be restarted without intervention, the master terminal operators of both subsystems are notified. This situation can require that the terminal user also be notified of the session's status and of any actions that must be taken, such as reentering the failing transaction or waiting until corrective action is taken by the MTO.

After an ISC transaction defined as recoverable is received and logged by IMS, it can be recovered across session and subsystem failures. After restart, when the session has been recovered, if IMS has pending output, it is sent as follows:

- When IMS sends output on the same session as that on which the input message is received, that output message is sent with the same type of headers as those originally sent with the input. Thus, a reply to a message sent with SEND/RECEIVE or SEND LAST is returned with the ATTACH FM header and a reply to a message sent with START/RETRIEVE is returned with the ATTACH and SCHEDULER FM headers.
- When IMS sends output on a session other than that on which the input message is received, that output message is considered as unsolicited output and is sent asynchronously (with ATTACH SCHEDULER).

After a restart, when IMS sends an output message, CICS reads the input message and initiates a transaction.

- Within the application, the EXEC ASSIGN STARTCODE parameter is examined to determine whether this transaction had been initiated synchronously or asynchronously.
- If the restart transaction is initiated asynchronously:
  - A RETRIEVE is issued to obtain the ISC reply. The EIB indicators should be saved. CICS attaches the end-user's terminal to this CICS program as the principal facility. This permits the IMS output to be issued directly to the terminal using a CICS EXEC SEND command.
  - The EIB indicators, saved after RETRIEVE execution, must be tested to determine whether SYNCPOINT or RETURN can be issued to end this transaction.
- If the transaction is initiated synchronously by SEND/RECEIVE or asynchronously by SEND LAST:
  - The input message is obtained using RECEIVE. In this case, the session is now the principal facility. Therefore, the RECEIVE command does not require specification of the SESSION parameter.
  - After the RECEIVE, the CICS application saves and then checks the EIB fields EIBSYNC, EIBFREE, and EIBRECV, in that order.
  - The application now issues an EXTRACT ATTACH to determine the format of the input data and the ID of the terminal that originally submitted the transaction.
  - The restart logic must now START an additional CICS transaction to acquire the end user's terminal as the principal facility. The START request is made with the terminal ID found in the PRN field by using the EXTRACT ATTACH as the TERMID parameter on the START.
  - The EIB indicators are now tested to determine whether a sync point is required and whether the session can be freed. If this can be done, the application can issue a RETURN command to implicitly cause a sync point to be issued and the session to be freed.

Within the context of the restart transaction, data can be retrieved for inquiry purposes or databases and files can be updated.

In addition to the foregoing logic, a CICS restart transaction can contain logic to:

- Store input received from the originating terminal on temporary storage. If it is necessary to re-create an input transaction, this information can be obtained and re-sent to IMS.
- Create logic to inform the terminal user of any special processing to be performed, such as reentering a transaction or waiting for some master terminal operator action before proceeding.

Chapter 35. ISC data flow control examples

The following topics provide examples of ISC data flow control.

Non-MFS bracket and half-duplex protocol examples

The following series of examples illustrate ISC data flow control protocols.

In the examples, the items in parentheses are optional on the flow.

For simplicity, the examples in this topic assume only-in-chain SCHEDULER messages for flows in both directions, and the response, ATTACH, and SCHEDULER protocols have been excluded. Also, the bracket protocol is considered symmetrical in that either the primary half session (PHS) or the secondary half session (SHS) can initiate the bracket as illustrated.

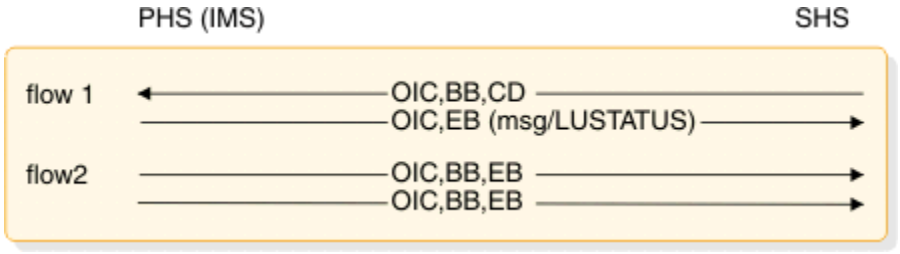


Figure 90. Example of bracket protocol for a PHS component defined to IMS as SINGLE1

Sample flow 1 shows the PHS ending a bracket on the first output following receipt of change-direction. When no output exists and the input cannot guarantee output, the bracket is ended through a stand-alone LUSTATUS.

Sample flow 2 shows the PHS sending output that occurs while in a between-brackets state.

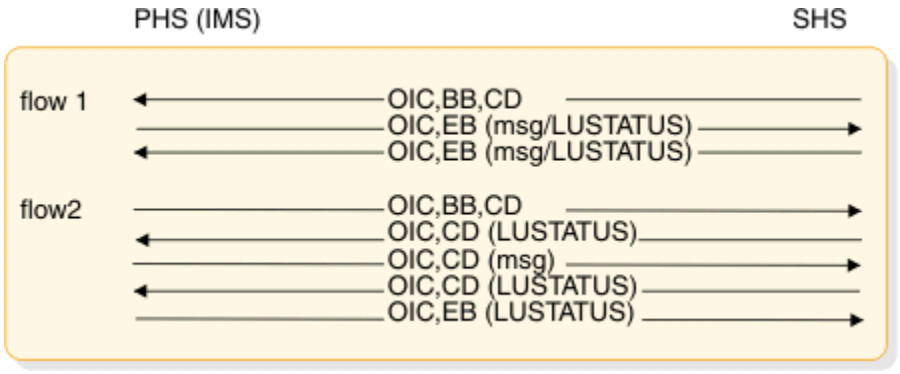


Figure 91. Example of bracket protocol for a PHS component defined to IMS as SINGLE2

Sample flow 1 shows the PHS returning the flow to a bracket initiator after receipt of a change-direction. The bracket initiator can optionally continue input or end the bracket. When no output exists and the input cannot guarantee output, the flow is returned to the bracket initiator using a stand-alone LUSTATUS.

Sample flow 2 shows the PHS sending asynchronous output. Also, IMS detects a potential LUSTATUS CD loop and forces end-bracket using LUSTATUS.

© Copyright IBM Corp. 1974, 2020

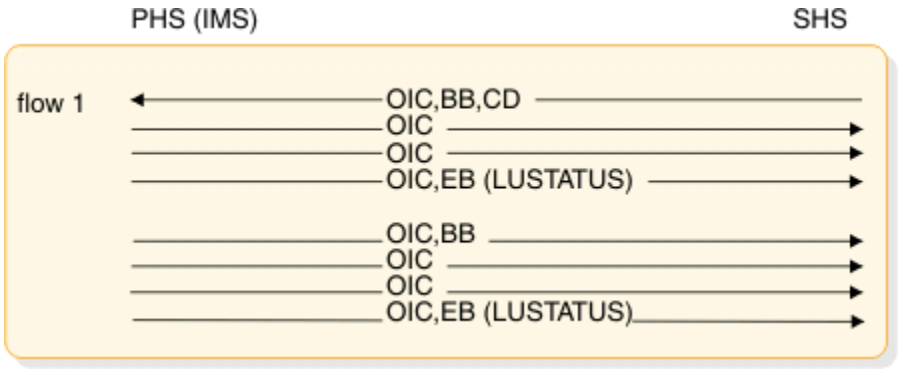


Figure 92. Example of bracket protocol for a PHS component defined to IMS as MULT1

Sample flow 1 shows the PHS ending a bracket using an LUSTATUS. The bracket ends following the last output from a queue after receipt of change-direction. If no output exists and the input cannot guarantee output, the bracket is ended immediately by using a stand-alone LUSTATUS.

Sample flow 2 shows the PHS both beginning and ending a bracket by using an LUSTATUS following the last output from a queue. Additional output available from other queues causes subsequent brackets to be initiated.

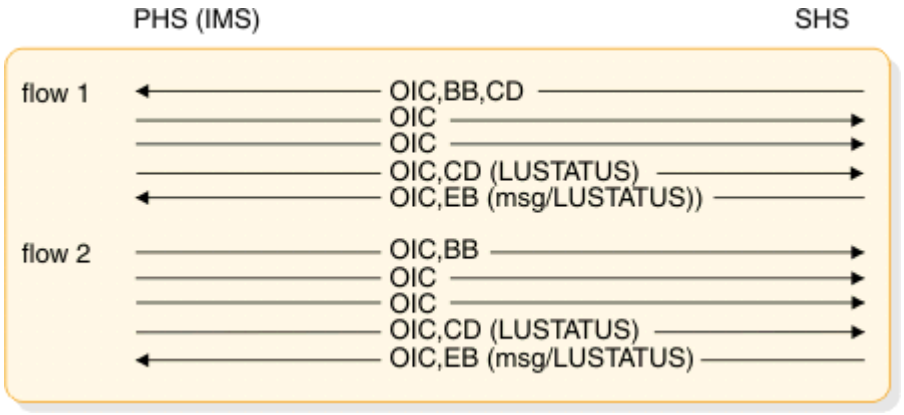


Figure 93. Example of bracket protocol for a PHS component defined to IMS as MULT2

Sample flow 1 shows the PHS returning the flow to the bracket initiator using an LUSTATUS. The bracket ends following the last output from a queue after receipt of a change-direction. The bracket initiator can optionally continue input or end the bracket. If no output exists and the input cannot guarantee output (that is, the input is not in conversational or response mode), the flow is returned by using a stand-alone LUSTATUS.

Sample flow 2 shows the PHS sending output that occurs in a between-brackets state.

MFS bracket and half-duplex protocol examples

The following topics provide examples for both MFS output and MFS input.

MFS output examples

The following figures provide examples of MFS output.

In the examples shown in the following figures, the IMS message consists of 3 presentation pages:

Logical page 1 that makes up 2 presentation pages
Logical page 2 that makes up 1 presentation page

In all of the examples, the ATTACH FM header showing DPN=X'.03' is optional.

Exception: For the first paging request, the ATTACH FM header showing DPN=X'03' is required.

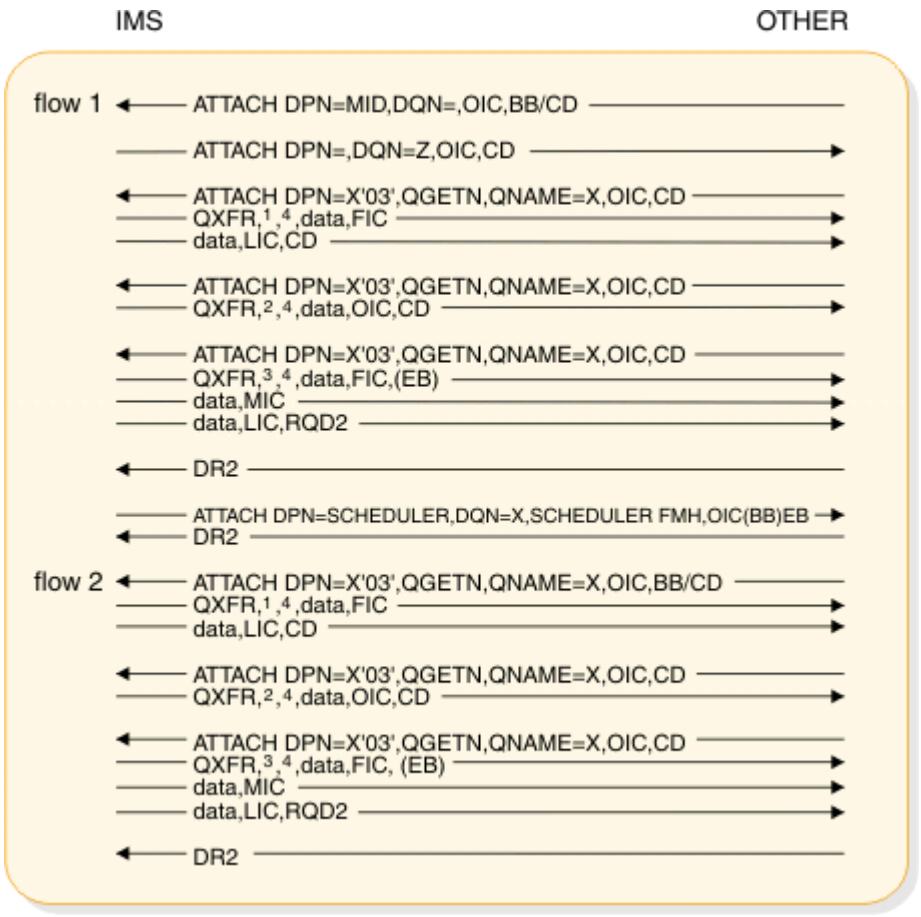


Figure 94. Example of demand-paged output, operator logical paging (OLP) not defined, sequential retrieval using QGETN FM header

IMS action: Remove message from the message queue

Note:

1. QCURSOR=0400010001, QCOUNT=020002
2. QCURSOR=0400010002, QCOUNT=020002
3. QCURSOR=0400020001, QCOUNT not present
4. DD FM header can precede the data

   The QNAME returned on paging requests must be the same value as sent on the output ATTACH DQN.

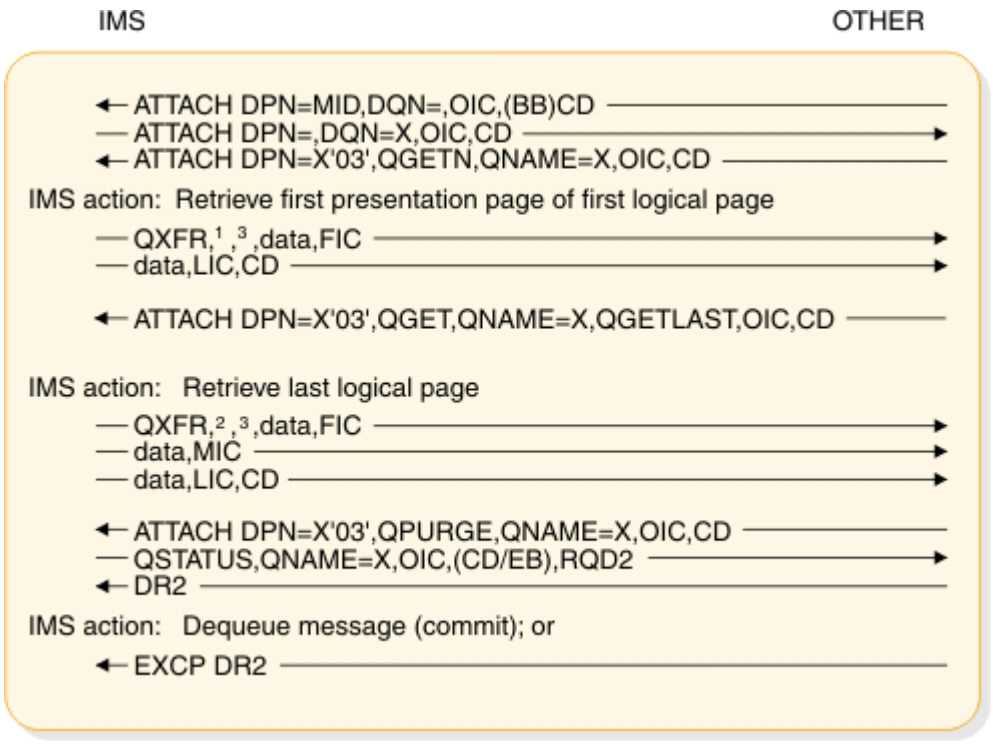


Figure 95. Example of demand-paged output, OLP defined, QGET (last page request and by cursor) used. QPURGE used to dequeue the message

Note:

1. QCURSOR=0400010001, QCOUNT=020002
2. CURSOR=0400020001, QCOUNT not present
3. Data descriptor FM header can precede the data

These same conditions for non-SCHEDULER demand-paged output result in an exception response and in subsequent ERP FMH7 being sent.

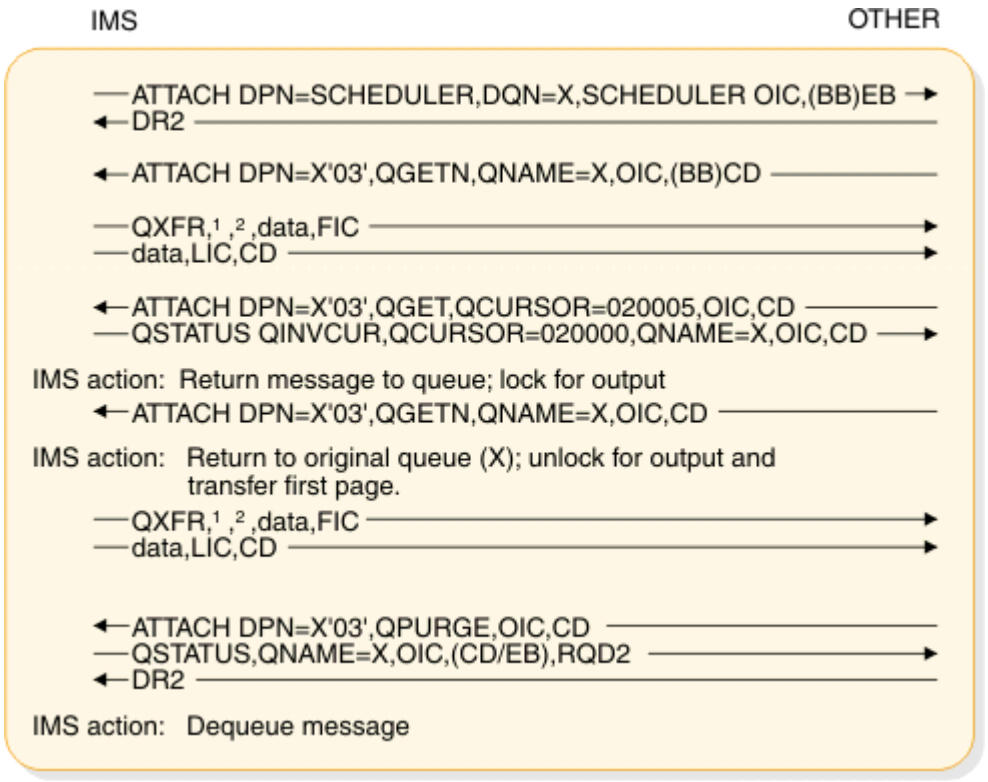


Figure 96. Example of demand-paged SCHEDULER output, OLP defined, QGET by cursor request for a page not within the range of output message

Note:

1. QCURSOR=0400010001, QCOUNT=020002
2. Data descriptor FM header can precede data

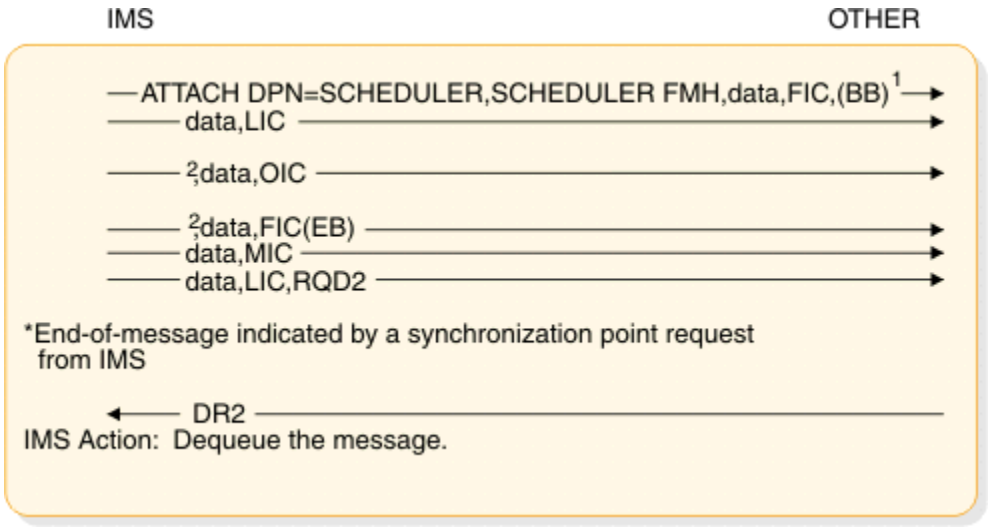


Figure 97. Example of autopaged output

Note:

1. Sending a first-in-chain with BB on autopaged output requests DR1 (RQD1).
2. Data descriptor FM header can precede the data in each chain. The ATTACH indicates a multichain message.

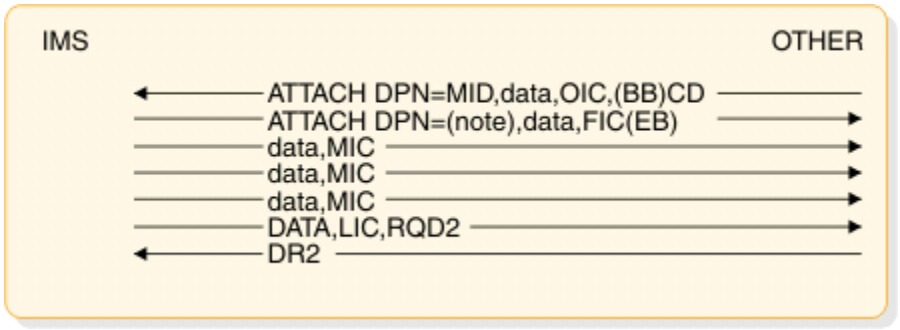


Figure 98. Example of nonpaged output message

Note: Data descriptor FM header can precede the data.

MFS input examples

The following figures provide examples of MFS input.

In the following examples, three DPAGEs are defined:

DPAGE 1 defines data for 2 segments
DPAGE 2 defines data for 1 segment
DPAGE 3 defines data for 3 segments

Three LPAGEs are created for the message. A data descriptor function management header with a DSN supplied with each chain is used to select each DPAGE. The first DSN name selects DPAGE 1, the second DSN name selects DPAGE 2, and the third DSN name selects DPAGE 3.

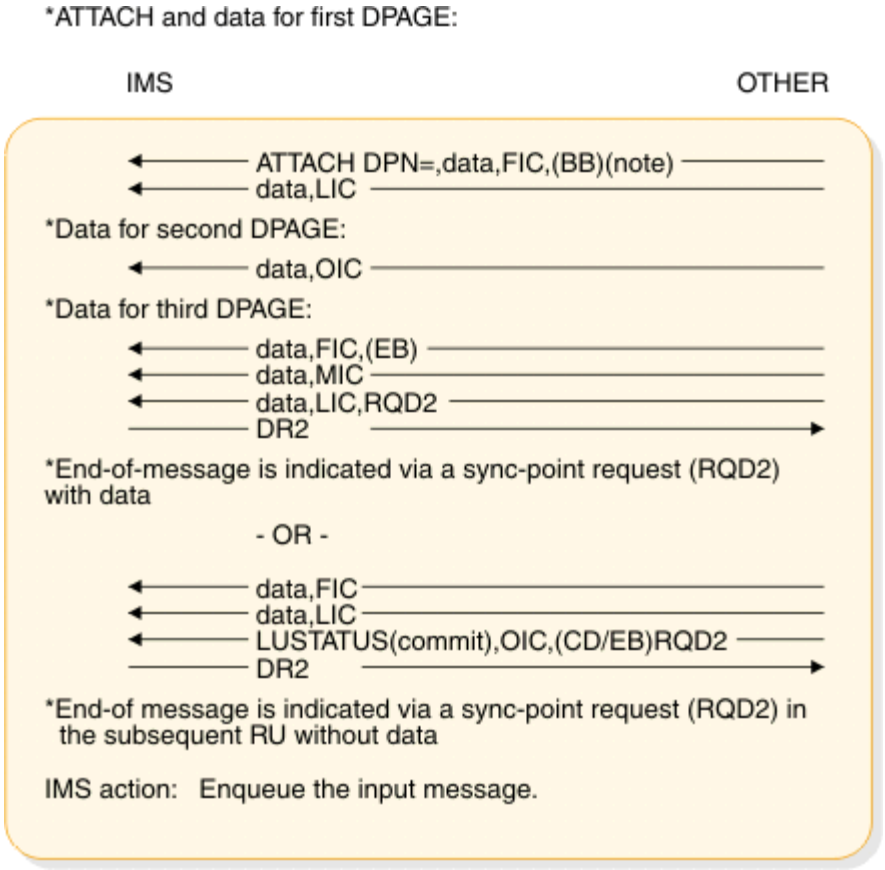


Figure 99. Example of autopaged input, three chains

A data descriptor FM header can precede the data.

Three LPAGEs are created for the message. A conditional test on the data is performed to select each DPAGE. Results of first test selects DPAGE 1, results of the second test selects DPAGE 2, and results of the third test selects DPAGE 3.

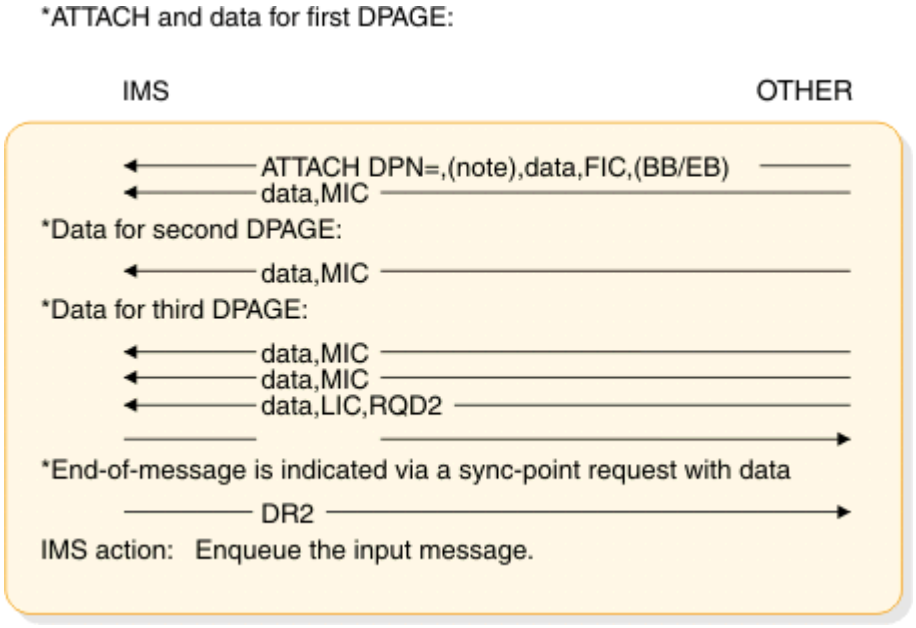


Figure 100. Example of autopaged input, single transmission chain

The data descriptor FM header can be sent to select one DPAGE only. The ATTACH indicates a multichain message.

SBI/BIS examples

IMS can be either the primary half session (PHS) or the secondary half session (SHS) for all of the examples given. Therefore, all of the functions and commands shown for PHS and SHS are IMS functions and commands.

Although all SBI and BIS sequences are valid for any session using LU 6.1 protocols, the following examples assume IMS to be both the primary and the secondary half session. No assumptions are made here as to when or how a subsystem other than IMS might send or receive these sequences.

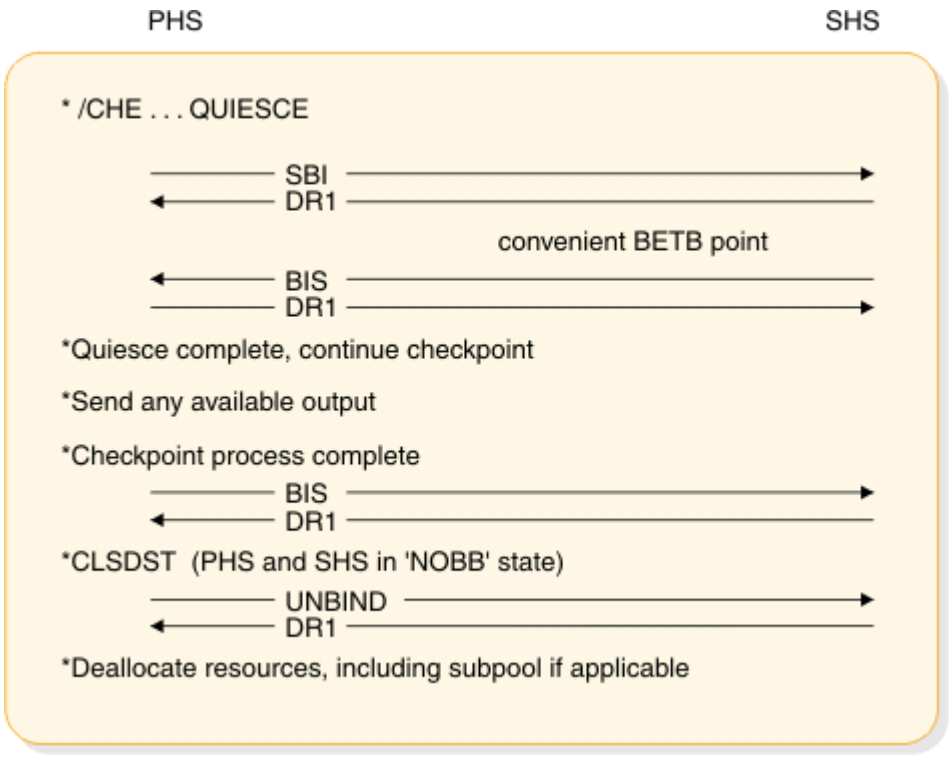


Figure 101. Example of PHS shutdown using SBI/BIS

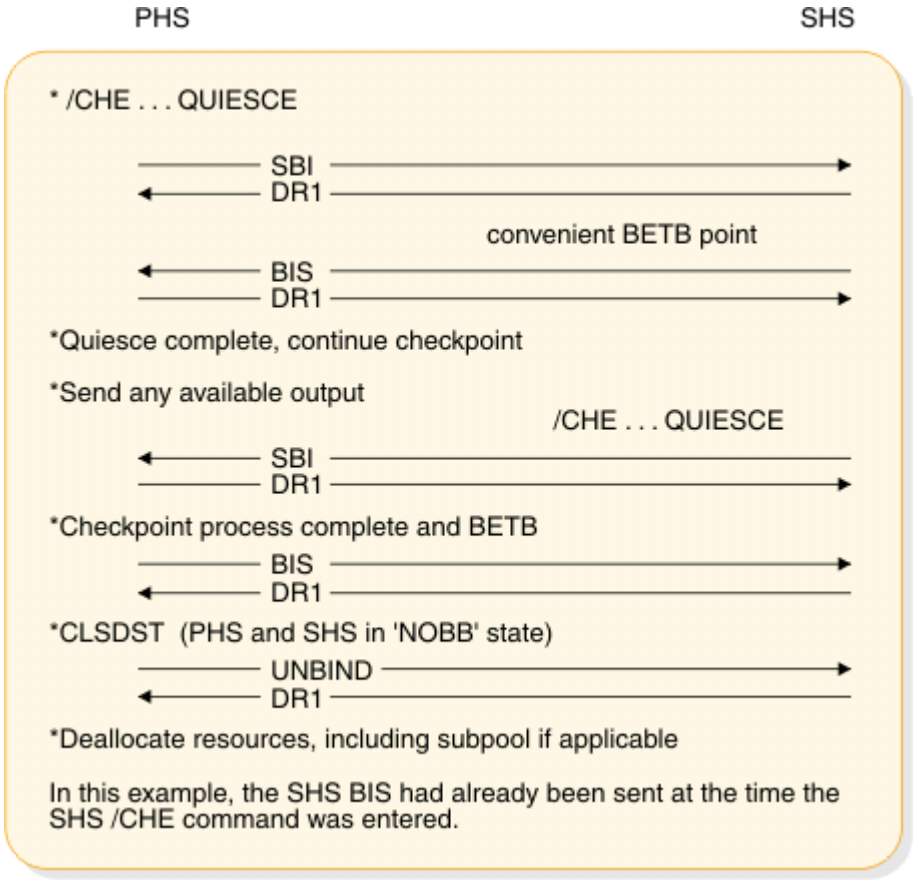


Figure 102. Example of nonsimultaneous shutdown using SBI/BIS by both half sessions

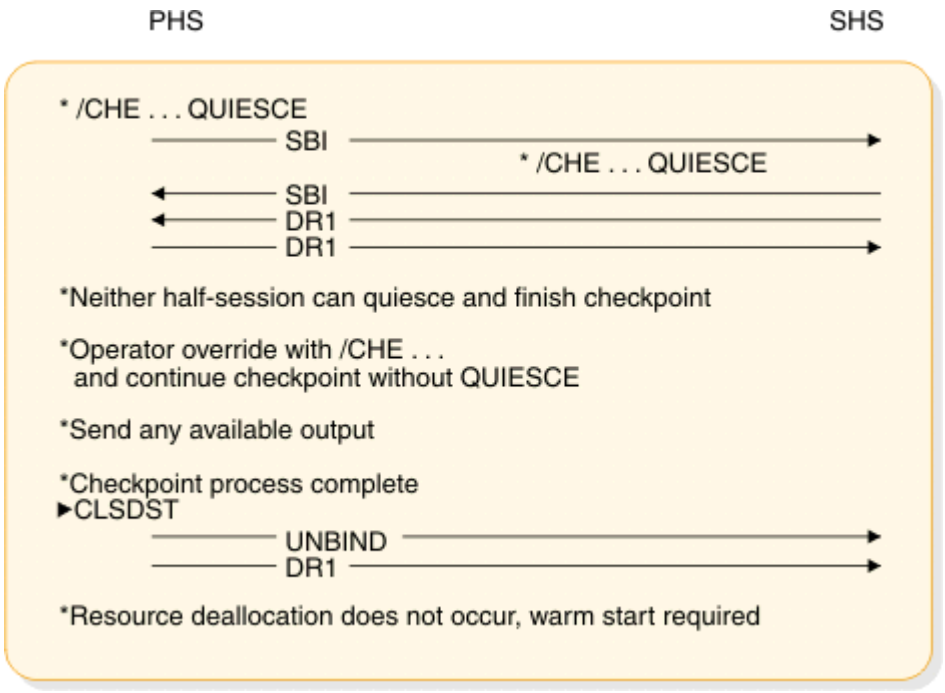


Figure 103. Example of simultaneous PHS and SHS shutdown using SBI/BIS fails because of operator override

Signal protocol example

The following example illustrates the use of SIGNAL RCD.

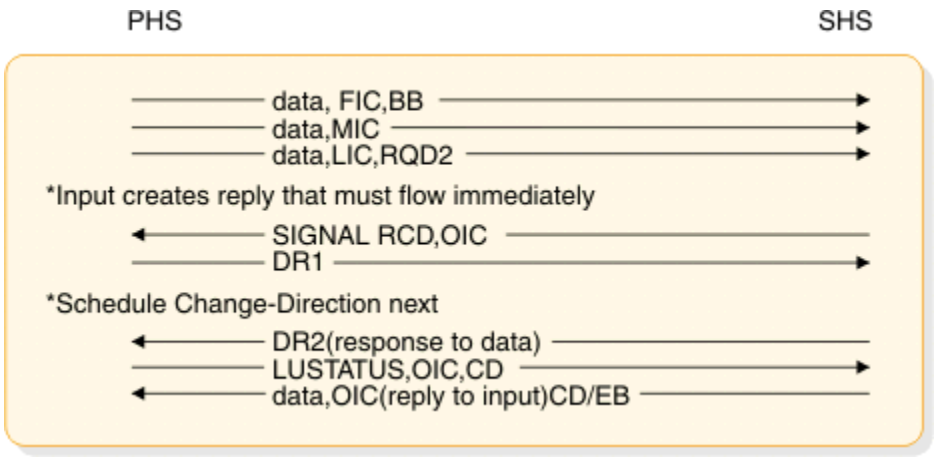


Figure 104. Signal protocol example

Chapter 36. ISC error recovery procedure examples

The following topics provide examples of the ISC error recovery procedure.

Sender-detected error examples

The following figures show examples of sender-detected errors during paged output (or input). IMS is the sender in both figures.

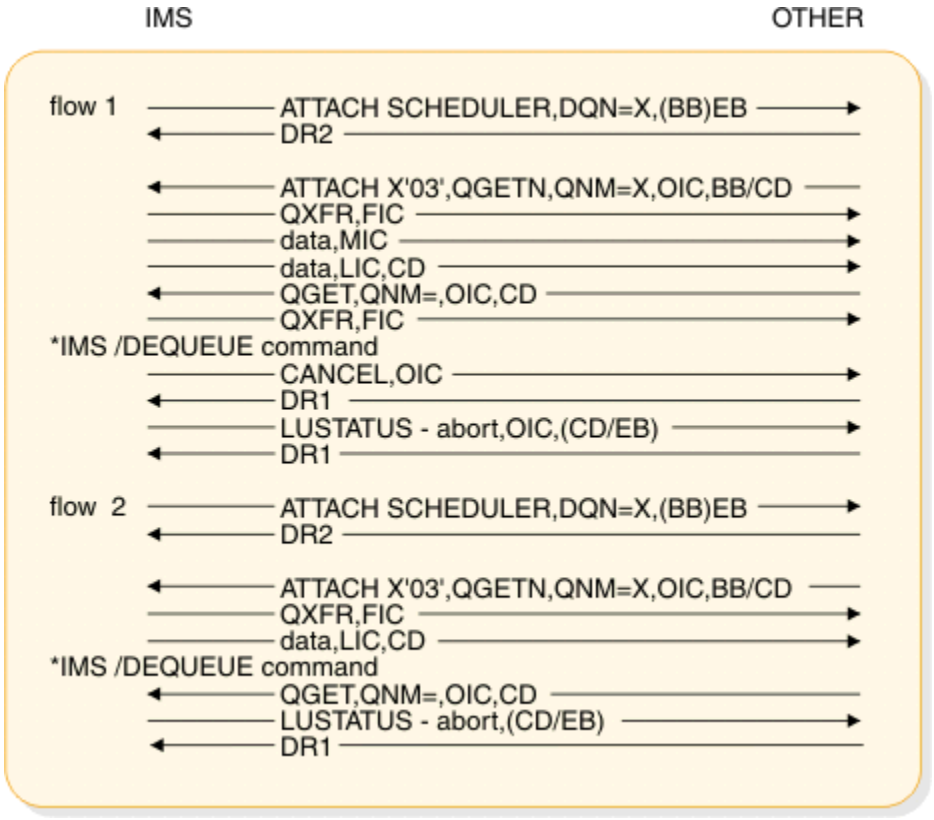


Figure 105. Example of sender-detected error while sending demand-paged output message

Flow 1 in Figure 96 shows an IMS /DEQUEUE command occurring during the second output demand page. The CANCEL command terminates the chain (page) in progress, and the LUSTATUS terminates the receiving process. If the paged output message follows another input or output message that began a bracket, the BB is not sent on the first page OIC.

Sample flow 2 is the same, except that the IMS /DEQUEUE command occurs between output pages so that the CANCEL command is unnecessary.

© Copyright IBM Corp. 1974, 2020

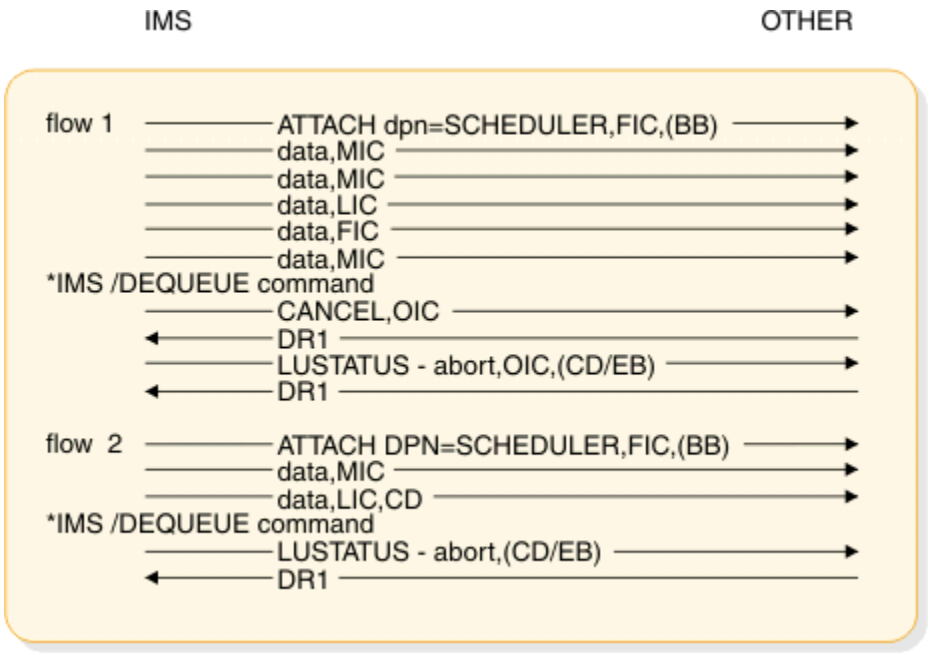


Figure 106. Example of sender-detected error while sending autopaged output message

Again, the CANCEL command terminates the chain (page) in progress and the LUSTATUS terminates the receiving process. The initial begin-bracket and the resulting bracket send/receive protocol are the same as for demand-paged output in Figure 105 on page 653.

Flow 2 is the same as flow 1 in Figure 97, except that the IMS /DEQUEUE command occurs while between output pages, so that the CANCEL command is unnecessary.

Receiver-detected error examples

For simplicity, in the following series of figures, all examples assume only-in-chain (OIC) messages for flows in both directions. Also the bracket and ERP protocol are considered symmetrical in that either half session can initiate the bracket or ERP procedures as illustrated.

For all session terminations not involving system failures, ATTACH information is recovered and backed up to the last sync point. In the examples that follow, IMS is the receiver.

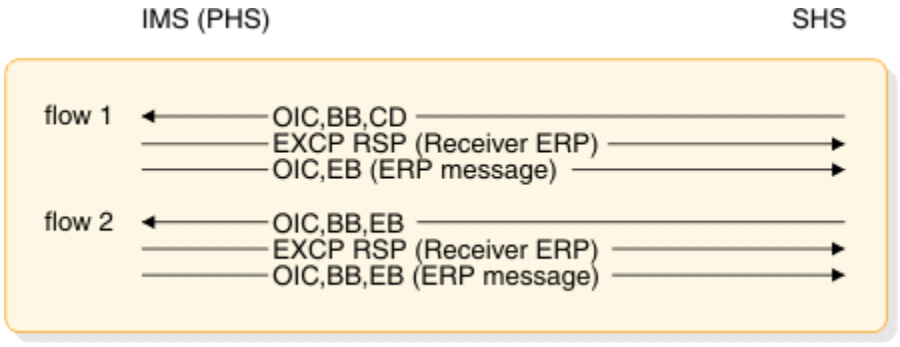


Figure 107. Example of receiver ERP for an IMS component defined as SINGLE1

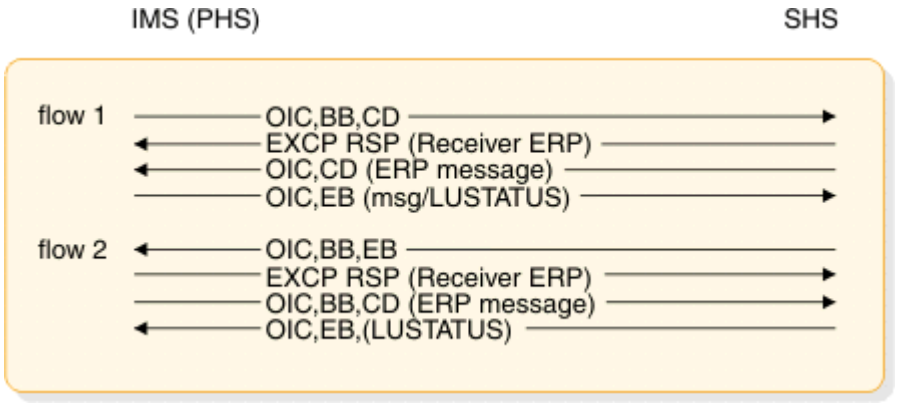


Figure 108. Example of receiver ERP for an IMS component defined as SINGLE2

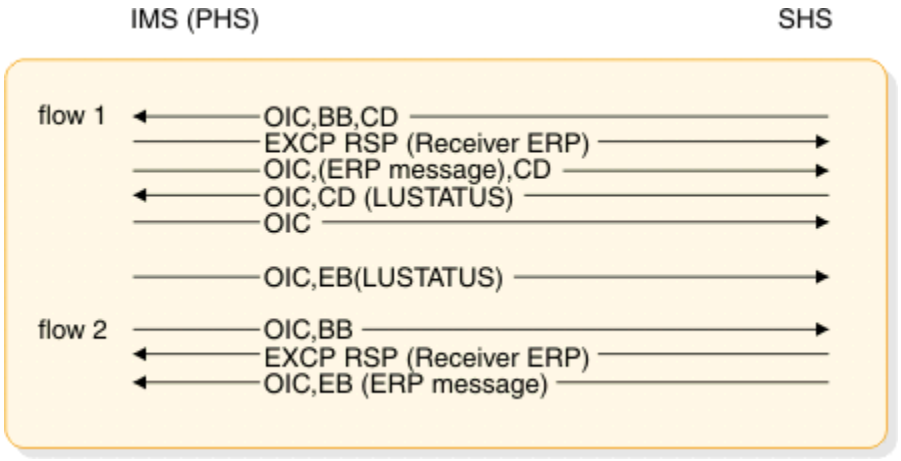


Figure 109. Example of receiver ERP for an IMS component defined as MULT1

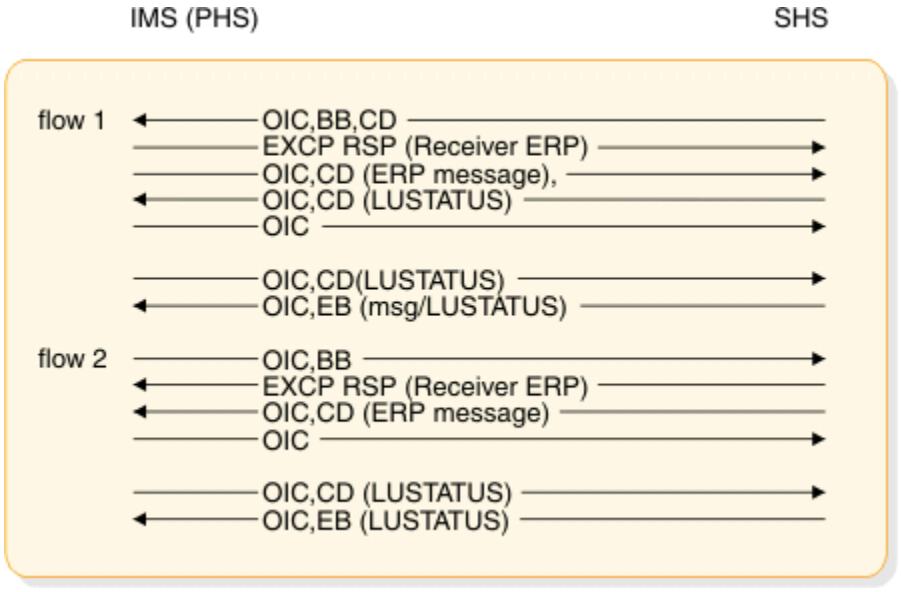


Figure 110. Example of receiver ERP for an IMS component defined as MULT2

In this case, the receiver is the other subsystem. The IMS action follows selective receiver ERP sequence.

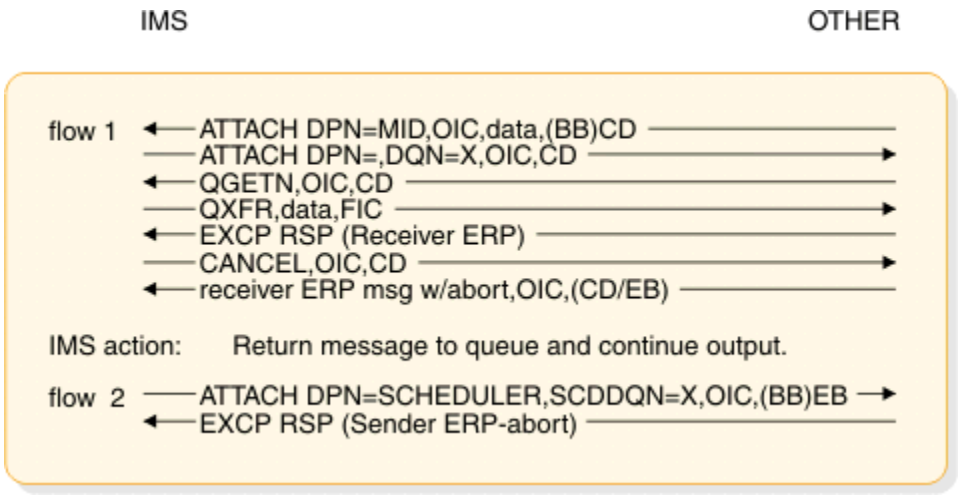


Figure 111. Example of demand-paged output with receiver-detected error

IMS action: Return message to queue and continue output.

In this example, flow 1 shows the receiver using receiver ERP to reject a demand-paged message. Flow 2 shows alternate methods with the same end result, but not involving an ERP message from the receiver.

Chapter 37. Sample program for IMS-CICS ISC

The following topics provide a sample program that illustrates the use of ISC between IMS and CICS.

By combining the sample program with the corresponding sample program in the *CICS Transaction Server for z/OS CICS Intercommunication Guide*, you can use the ISC function in both products.

The sample program is shipped as object and source code data set DFSISC00. The load library containing the object code is IMS.ADFSLOAD. The source library containing the source code is IMS.ADFSSRC.

Installation procedure

Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

The steps you must complete include:

1. Compile and bind the sample COBOL program.
2. Define the transaction codes to IMS using the system definition procedure.
3. Define an ISC session between IMS and CICS using the system definition procedure.
4. Compile the MFS utility statements to place the necessary formats into the online library.
5. Perform the PSB generation and then the ACB generation.
6. Initialize the IMS system.
7. Establish a session between IMS and CICS.
8. Run the corresponding CICS sample program.

In related topics, sample code is provided for performing steps 1 through 5.

Related reference

IMS sample program (DFSISC00)
The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

Job control statements for the sample program
Use the following JCL to compile and bind the sample program.

IMS system definition statements
Use the following statements to define to IMS the two transaction codes supported by the sample program.

MFS formats
The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

Program specification block (PSB) generation for the sample program
The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

Application control block (ACB) generation

© Copyright IBM Corp. 1974, 2020

Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

IMS sample program (DFSISC00)

The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

The program has two transaction codes: SAMPLA1 and SAMPLA2. When invoked using SAMPLA1, the output message is not formatted by MFS. When invoked using SAMPLA2, the output message is formatted by MFS using distributed presentation management (DPM).

```
CBL APOST
       IDENTIFICATION DIVISION.
           PROGRAM-ID. SAMPLA.
           AUTHOR:
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  IBM-370.
       OBJECT-COMPUTER.  IBM-370.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
       DATA DIVISION.
       FILE SECTION.
           EJECT
       WORKING-STORAGE SECTION.
       77  BEGIN-LIT               PIC X(16) VALUE 'BEGIN 77 ENTRIES'.
       77  FILLER                      PIC X(45) VALUE
            '****DATE LAST COMPILED: xxx xx, 19xx****'.
       77  CALL-FUNCTION               PIC XXXX.
       77  MOD-NAME                    PIC X(08).
           EJECT
       01  INPUT-AREA.
           02  IN-LL                   PIC S9999 COMP.
           02  IN-ZZ                   PIC S999 COMP.
           02  TRAN-CODE               PIC X(08).
           02  INPUT-DATA              PIC X(79).
           02  INPUT-DATA-D            REDEFINES INPUT-DATA.
               05  INPUT-DATA-1        OCCURS 3 TIMES
                                       PIC X(20).
               05  INPUT-DATA-2        PIC X(19).
       01  OUTPUT-AREA.
           02  OUT-LL                  PIC S9999 COMP.
           02  OUT-ZZ                  PIC S999 COMP.
           02  OUTPUT-DATA             PIC X(79).
       01  OUTPUT-AREA-1.
           02  OUT-LL-1                PIC S9999 COMP.
           02  OUT-ZZ-1                PIC S999 COMP.
           02  OUTPUT-DATA-1           PIC X(20).
           EJECT
       01  DLI-FUNCTIONS.
           02  GET-UNIQ                PIC XXXX VALUE 'GU  '.
           02  GET-NEXT                PIC XXXX VALUE 'GN  '.
           02  ISRT                    PIC XXXX VALUE 'ISRT'.
           EJECT
       01  STATUS-ERROR-SEG.
           02  FILLER                  PIC S999 COMP VALUE +83.
           02  FILLER                  PIC S999 COMP VALUE +0.
           02  FILLER                  PIC X(28)  VALUE
               '******* STATUS ERROR *******'.
           02  FILLER                  PIC X(10) VALUE ' TRANCODE:'.
           02  ERROR-TRAN              PIC X(8).
           02  FILLER                  PIC X(17)   VALUE
                  ' STATUS RECEIVED:'.
           02  ERROR-STATUS            PIC XX.
           02  FILLER                  PIC X(10) VALUE ' FUNCTION:'.
           02  ERROR-FUNCTION          PIC XXXX.
           EJECT
       LINKAGE SECTION.
       01  IOTP-PCB.
           02  IOTP-LTERM              PIC X(8).
           02  FILLER                  PIC XX.
           02  IOTP-STATUS             PIC XX.
           02  IOTP-PREFIX.
               03  IOTP-DATE           PIC S9(7) COMP-3.
               03  IOTP-TIME           PIC S9(7) COMP-3.
               03  IOTP-MSG-NUMBER     PIC S999 COMP.
```

```
           03  FILLER              PIC XX.
       02  IOTP-MOD-NAME           PIC X(8).
       EJECT

 PROCEDURE DIVISION.
       ENTRY 'DLITCBL' USING IOTP-PCB.
 100-RETRIEVE-MESSAGE-SEGMENT.
       MOVE GET-UNIQ TO CALL-FUNCTION.
       CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB INPUT-AREA.
       IF IOTP-STATUS = 'QC'
           GO TO 800-GOBACK-ROUTINE.
       IF IOTP-STATUS NOT = SPACES
           GO TO 700-INVALID-STATUS-CODE.
 200-CHECK-TRAN-CODE.
      ***************************************************************
      *    THE ONLY DIFFERENCE BETWEEN THESE TWO TRANSACTIONS       *
      *    IS THE ABSENCE OR PRESENCE OF MFS.  SAMPLA1 DOES NOT     *
      *    CONTAIN MFS.  SAMPLA2 CONTAINS MFS.                      *
      ***************************************************************
       IF TRAN-CODE = 'SAMPLA2'
           PERFORM  400-SAMPLA2-ROUTINE
               THRU 450-SAMPLA2-ROUTINE-EXIT,
       ELSE
           PERFORM 300-SAMPLA1-ROUTINE
               THRU 350-SAMPLA1-ROUTINE-EXIT.
       GO TO 100-RETRIEVE-MESSAGE-SEGMENT.
       EJECT

 300-SAMPLA1-ROUTINE.
       MOVE ISRT TO CALL-FUNCTION.
       SUBTRACT 8 FROM IN-LL GIVING OUT-LL.
       MOVE IN-ZZ TO OUT-ZZ.
       MOVE INPUT-DATA TO OUTPUT-DATA.
       CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB OUTPUT-AREA.
       IF IOTP-STATUS NOT = SPACES
           GO TO 700-INVALID-STATUS-CODE.
 350-SAMPLA1-ROUTINE-EXIT.
       EXIT.
 400-SAMPLA2-ROUTINE.
       MOVE 'MODA' TO MOD-NAME,
       MOVE +0 TO OUT-ZZ-1.
       MOVE 24 TO OUT-LL-1.
       MOVE INPUT-DATA-1 (1) TO OUTPUT-DATA-1.
       PERFORM 500-INSERT-ROUTINE
          THRU 550-INSERT-ROUTINE-EXIT.
       SUBTRACT 20 FROM IN-LL.
       IF IN-LL IS LESS THAN 13 GO TO 450-SAMPLA2-ROUTINE-EXIT.
       MOVE INPUT-DATA-1 (2) TO OUTPUT-DATA-1.
       PERFORM 600-INSERT-ROUTINE
          THRU 650-INSERT-ROUTINE-EXIT.
       SUBTRACT 20 FROM IN-LL.
       IF IN-LL IS LESS THAN 13 GO TO 450-SAMPLA2-ROUTINE-EXIT.
       MOVE INPUT-DATA-1 (3) TO OUTPUT-DATA-1.
       PERFORM 600-INSERT-ROUTINE
          THRU 650-INSERT-ROUTINE-EXIT.
       SUBTRACT 20 FROM IN-LL.
       IF IN-LL IS LESS THAN 13 GO TO 450-SAMPLA2-ROUTINE-EXIT.
       MOVE INPUT-DATA-2 TO OUTPUT-DATA-1.
       PERFORM 600-INSERT-ROUTINE
 450-SAMPLA2-ROUTINE-EXIT.
          THRU 650-INSERT-ROUTINE-EXIT.
       EXIT.
       EJECT
 500-INSERT-ROUTINE.
       MOVE ISRT TO CALL-FUNCTION.
       CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB OUTPUT-AREA-1
                            MOD-NAME.
       IF IOTP-STATUS NOT = SPACES
           GO TO 700-INVALID-STATUS-CODE.
 550-INSERT-ROUTINE-EXIT.
       EXIT.
 600-INSERT-ROUTINE.
       MOVE ISRT TO CALL-FUNCTION.
       CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB OUTPUT-AREA-1.
       IF IOTP-STATUS NOT = SPACES
           GO TO 700-INVALID-STATUS-CODE.
 650-INSERT-ROUTINE-EXIT.
       EXIT.
 700-INVALID-STATUS-CODE.
       MOVE CALL-FUNCTION TO ERROR-FUNCTION.
       MOVE IOTP-STATUS TO ERROR-STATUS.
       MOVE TRAN-CODE TO ERROR-TRAN.
```

```
           MOVE ISRT TO CALL-FUNCTION.
           CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB STATUS-ERROR-SEG.
       800-GOBACK-ROUTINE.
           GOBACK.
```

Related tasks

Installation procedure
Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

Related reference

Job control statements for the sample program
Use the following JCL to compile and bind the sample program.

IMS system definition statements
Use the following statements to define to IMS the two transaction codes supported by the sample program.

MFS formats
The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

Program specification block (PSB) generation for the sample program
The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

Application control block (ACB) generation
Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

Job control statements for the sample program

Use the following JCL to compile and bind the sample program.

This JCL uses the z/OS COBOL 2.4 compiler and places the resulting load module in IMS.PGMLIB.

```
//IMSCOBOL JOB ACCT,NAME,CLASS=A,MSGLEVEL=(1,1),MSGCLASS=A
//       PROC  MBR=,PAGES=60,RGN=512K,
//             SOUT=A
//C      EXEC  PGM=IKFCBL00,REGION=&RGN,
//             PARM='SIZE=130K,BUF=10K,LINECNT=50,APOST,BATCH'
//SYSLIN DD    DSN=&&LIN,DISP=(MOD,PASS),UNIT=SYSDA,
//             DCB=(USER.PROCLIB),
//             SPACE=(3520,(40,10),RLSE,,ROUND)
//SYSPRINT DD  SYSOUT=&SOUT,DCB=(LRECL=121,BLKSIZE=605,RECFM=FBA),
//             SPACE=(605,(&PAGES.0,&PAGES),RLSE,,ROUND)
//SYSUT1 DD    UNIT=SYSDA,DISP=(,DELETE),
//             SPACE=(3520,(100,10),RLSE,,ROUND)
//SYSUT2 DD    UNIT=SYSDA,DISP=(,DELETE),
//             SPACE=(3520,(100,10),RLSE,,ROUND)
//SYSUT3 DD    UNIT=SYSDA,DISP=(,DELETE),
//             SPACE=(3520,(100,10),RLSE,,ROUND)
//SYSUT4 DD    UNIT=SYSDA,DISP=(,DELETE),
//             SPACE=(3520,(100,10),RLSE,,ROUND)
//L      EXEC  PGM=IEWL,REGION=&RGN,PARM='XREF,LET,LIST',
//             COND=(4,LT,C)
//*STEPLIB DD   DSN=IMS.SDFSRESL,DISP=SHR
//SYSLIB DD    DSN=SYS1.COBLIB,DISP=SHR
//SDFSRESL DD    DSN=IMS.SDFSRESL,DISP=SHR
//SYSLIN DD    DSN=&&LIN,DISP=(OLD,DELETE),VOL=REF=*.C.SYSLIN
//       DD    DSN=IMS.PROCLIB(CBLTDLI),DISP=SHR
//       DD    DDNAME=SYSIN
//SYSLMOD DD   DSN=IMS.PGMLIB(&MBR),DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605),
//             SPACE=(605,(&PAGES.0,&PAGES),RLSE,,ROUND)
//SYSUT1 DD    UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),DISP=(,DELETE),
//             SPACE=(3520,(100,10),RLSE,,ROUND)
//   PEND
//IMSCOBOL EXEC IMSCOBOL,SOUT=A,MBR=SAMPLA
//C.SYSIN DD *
```

You can also bind the sample program directly from IMS.ADFSLOAD to IMS.PGMLIB.

Related tasks

Installation procedure
Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

Related reference

IMS sample program (DFSISC00)
The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

IMS system definition statements
Use the following statements to define to IMS the two transaction codes supported by the sample program.

MFS formats
The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

Program specification block (PSB) generation for the sample program
The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

Application control block (ACB) generation
Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

IMS system definition statements

Use the following statements to define to IMS the two transaction codes supported by the sample program.

Restriction: The DPM-Bn specified in the following code must match the DPM-Bn specified on the MFS format DEV statement.

```
   APPLCTN  PSB=SAMPLA
      TRANSACT CODE=SAMPLA1,INQ=(YES,NORECOV)
      TRANSACT CODE=SAMPLA2,INQ=NO
   TYPE  UNITYPE=LUTYPE6,OPTIONS=(TRANRESP,OPNDST,NOMTOMSG,            X
               SYNCSESS),                                              X
               MSGDEL=SYSINFO,                                         X
               OUTBUF=256,                                             X
               SEGSIZE=256
*
**CICSA1 PARALLEL SESSION NODE
*
         TERMINAL  NAME=CICSA1,                                        X
               SESSION=5,                                              X
               COMPT1=(SINGLE1,DPM-B1,IGNORE),                         X
               COMPT2=(SINGLE2,DPM-B1,IGNORE),                         X
               COMPT3=(MULT1,DPM-B1,IGNORE),                           X
               COMPT4=(MULT2,DPM-B1,IGNORE)
    VTAMPOOL
*
         SUBPOOL  NAME=PS01
            NAME  PSLT01,COMPT=1,ICOMPT=1
            NAME  PSLT02,COMPT=2,ICOMPT=2
            NAME  PSLT03,COMPT=3,ICOMPT=3
            NAME  PSLT04,COMPT=4,ICOMPT=4
```

Related tasks

Installation procedure
Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

Related reference

IMS sample program (DFSISC00)

The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

Job control statements for the sample program
Use the following JCL to compile and bind the sample program.

MFS formats
The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

Program specification block (PSB) generation for the sample program
The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

Application control block (ACB) generation
Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

MFS formats

The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

This is shipped as copy code in data set MFSSISC0 in IMS.ADFSMAC.

```
  //MFSUTL JOB ACCT,NAME,CLASS=A,MSGLEVEL=(1,1)
  //JOBLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
  //         PROC RGN=360K,SOUT=A,SNODE=IMSVS,
  //             SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
  //             PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
  //             COMPR=NOCOMPRESS,COMPR2=COMPRESS,
  //             LN=55,SN=8,DEVCHAR=0
  //S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
  //  PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,&COMPR,
  //  'LINECNT=&LN,STOPRC=&SN,DEVCHAR=&DEVCHAR')
  //SYSLIB   DD   DSN=IMS.SDFSMAC,DISP=SHR
  //SYSIN    DD   DSN=&SNODE..&SOR.(&MBR),DISP=SHR
  //REFIN    DD   DSN=IMS.REFERAL,DISP=OLD
  //REFOUT   DD   DSN=IMS.REFERAL,DISP=OLD
  //REFRD    DD   DSN=IMS.REFERAL,DISP=OLD
  //SYSTEXT  DD   DSN=&&TXTPASS,UNIT=SYSDA,
  //             SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
  //SYSUT3   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
  //SYSUT4   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
  //UTPRINT  DD   SYSOUT=&SOUT
  //SYSPRINT DD   SYSOUT=&SOUT,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
  //SYSUDUMP DD   SYSOUT=&SOUT
  //SEQBLKS  DD   DSN=&&BLKS,DISP=(NEW,PASS),
  //             UNIT=SYSDA,SPACE=(CYL,(1,1))
  //S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
  //             PARM='&COMPR2,DEVCHAR=&DEVCHAR',
  //             COND=(8,LT,S1)
  //SEQBLKS  DD   DSN=&&BLKS,DISP=(OLD,DELETE)
  //UTPRINT  DD   SYSOUT=&SOUT,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
  //SYSUDUMP DD   SYSOUT=&SOUT
  //FORMAT   DD   DSN=IMS.FORMAT,DISP=OLD
  //SYSPRINT DD   SYSOUT=&SOUT
  //SYSUT3   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
  //SYSUT4   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
  //  PEND
  //MFSUTL   EXEC MFSUTL,SOUT=A
  //S1.SYSIN DD *
         PRINT ON,NOGEN
FMTA     FMT
         SPACE 3
         DEV   TYPE=DPM-B1,FEAT=IGNORE,DSCA=X'00A0'
         SPACE 2
         DIV   TYPE=OUTPUT,OPTIONS=(DPAGE,NODNM)
         SPACE 3
DPAGE01  DPAGE FILL=NULL
PPAGE01  PPAGE
         DFLD  'PAGE'
LPGNO    DFLD  LTH=04
         DFLD  ' '
         SPACE 2
```

```
DATA01   DFLD  LTH=20
         SPACE 3
         FMTEND
         EJECT
MODA     MSG   TYPE=OUTPUT,SOR=(FMTA,IGNORE),PAGE=YES
         SPACE 3
         SEG
         SPACE 2
         MFLD  (LPGNO,LPAGENO)
         MFLD  DATA01,LTH=20
         SPACE 3
         MSGEND
         END
/*
```

Related tasks

Installation procedure
Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

Related reference

IMS sample program (DFSISC00)
The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

Job control statements for the sample program
Use the following JCL to compile and bind the sample program.

IMS system definition statements
Use the following statements to define to IMS the two transaction codes supported by the sample program.

Program specification block (PSB) generation for the sample program
The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

Application control block (ACB) generation
Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

Program specification block (PSB) generation for the sample program

The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

```
//PSBGEN   EXEC PSBGEN,SOUT=A,MBR=SAMPLA
//C.SYSIN DD *
        PSBGEN   LANG=COBOL,PSBNAME=SAMPLA
        END
/*
```

Related tasks

Installation procedure
Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

Related reference

IMS sample program (DFSISC00)
The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

Job control statements for the sample program
Use the following JCL to compile and bind the sample program.

IMS system definition statements

Use the following statements to define to IMS the two transaction codes supported by the sample program.

MFS formats
The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

Application control block (ACB) generation
Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

Application control block (ACB) generation

Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library.

```
//ACBGEN JOB ACCT,NAME,CLASS=A,MSGLEVEL=(1,1)
//ACBGEN   EXEC ACBGEN,SOUT=A
//G.SYSIN DD *
 BUILD PSB=SAMPLA
 BUILD PSB=SAMPLB
/*
```

Related tasks

Installation procedure
Before the sample program can be used to perform ISC functions between IMS and CICS , you must complete several steps.

Related reference

IMS sample program (DFSISC00)
The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS.

Job control statements for the sample program
Use the following JCL to compile and bind the sample program.

IMS system definition statements
Use the following statements to define to IMS the two transaction codes supported by the sample program.

MFS formats
The following statements comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program.

Program specification block (PSB) generation for the sample program
The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program.

Part 8. Multiple Systems Coupling (MSC)

These topics introduce multiple systems coupling (MSC). You can use MSC to connect multiple IMS subsystems. These topics include an overview of MSC and the information you need to design, implement, and administer an MSC network.

© Copyright IBM Corp. 1974, 2020

Chapter 38. Overview of Multiple Systems Coupling

Multiple Systems Coupling (MSC) makes it possible for transactions to be entered in one IMS and processed in another IMS.

The responses can be returned to the terminals that entered the transactions or to other terminals. IMS uses MSC to route and control message traffic between connected IMS systems.

Multiple Systems Coupling concepts

MSC provides the ability to connect geographically dispersed IMS systems in such a way as to allow programs and operators of one IMS to access programs and operators of the connected IMS systems.

Communication can occur between two or more (up to 2036) IMS systems running on any supported combination of operating systems.

MSC also provides a way to extend the throughput of an IMS beyond the capacity of a single CPU. This extension is possible if the IMS applications can be partitioned among IMS systems two ways:

- *Horizontal partitioning*: Applications execute in more than one IMS with database contents split between IMS systems.
- *Vertical partitioning*: Applications execute in one IMS with the complete database that they reference attached to that IMS. Transactions can originate in any IMS.

A *link* is a connection between two IMS systems. All links must be defined during the IMS system definitions for each IMS. There are two types of links: physical links and logical links.

- A *physical link* is the access method connection or hardware connection between two IMS systems.
- A *logical link* is the mechanism through which a physical link is related to the transactions and terminals that make use of that physical link.

You can assign a logical link to a physical link during system definition or you can assign it dynamically by using the CREATE MSLINK command. You can also dynamically update the definition later by using either the IMS type-2 UPDATE MSLINK command or the type-1 /MSASSIGN LINK command.

MSC physical links

A physical link is how the IMS systems connect to one other through access methods or hardware.

You can define physical links in online IMS systems by using the **CREATE MSPLINK** command.

You can also define physical links during IMS system definition by using the MSPLINK stage-1 system definition macro.

To update physical links in online IMS systems, use the **UPDATE MSPLINK** command.

To save physical links that are created or updated by using type-2 commands across IMS cold starts, either export the physical link definitions to the IMSRSC repository or code the changes to the MSC resources into stage-1 system definition macros.

A maximum of 1018 physical links are allowed in each IMS in a Multiple Systems Coupling (MSC) network.

Multiple Systems Coupling supports the following types of physical links:

Channel-to-channel (CTC) adapter
Usually used only when the IMS systems are in the same data center.

The CTC adapter is a channel-to-channel hardware connection. You can assign only one logical link to a physical link that uses the CTC connection type.

Memory-to-memory (MTM)
Used when the IMS systems are in the same logical partition.

© Copyright IBM Corp. 1974, 2020

The MTM link is a software link between IMS subsystems that are running in the same logical partition. You can assign only one logical link to a physical link that uses the MTM connection type.

TCP/IP
Usually used when the IMS systems are in different data centers. IMS Connect manages the TCP/IP connections and protocols for the physical links.

The TCP/IP connection and networking protocols are the protocols that are used by the internet. IMS Connect manages the TCP/IP connections and protocols for MSC. Communications between MSC and IMS Connect are managed by the Structured Call Interface (SCI) component of the IMS Common Service Layer (CSL) in an IMSplex. You can assign multiple logical links to a physical link that uses the TCP/IP connection type. TCP/IP physical links always operate in MSC bandwidth mode and require a slightly larger buffer size than the other physical link types.

VTAM
Usually used when the IMS systems are in different data centers.

VTAM is an access method that usually uses a teleprocessing media connection. You can assign multiple logical links to a physical link that uses the VTAM connection type.

From an MSC perspective, the operation of TCP/IP physical links and VTAM physical links is similar. Apart from differences in system definition and buffer size requirements, the IMS and z/OS components that support each connection type present the most significant operational differences.

Both TCP/IP and VTAM physical link types can be used either as your primary physical link type, or as a backup link type in case the other link type fails for any reason.

Depending on various factors, such as network traffic and the distance between the two connected IMS systems, a TCP/IP physical link is likely to provide better performance than a VTAM physical link.

The following figure illustrates the types of physical links.

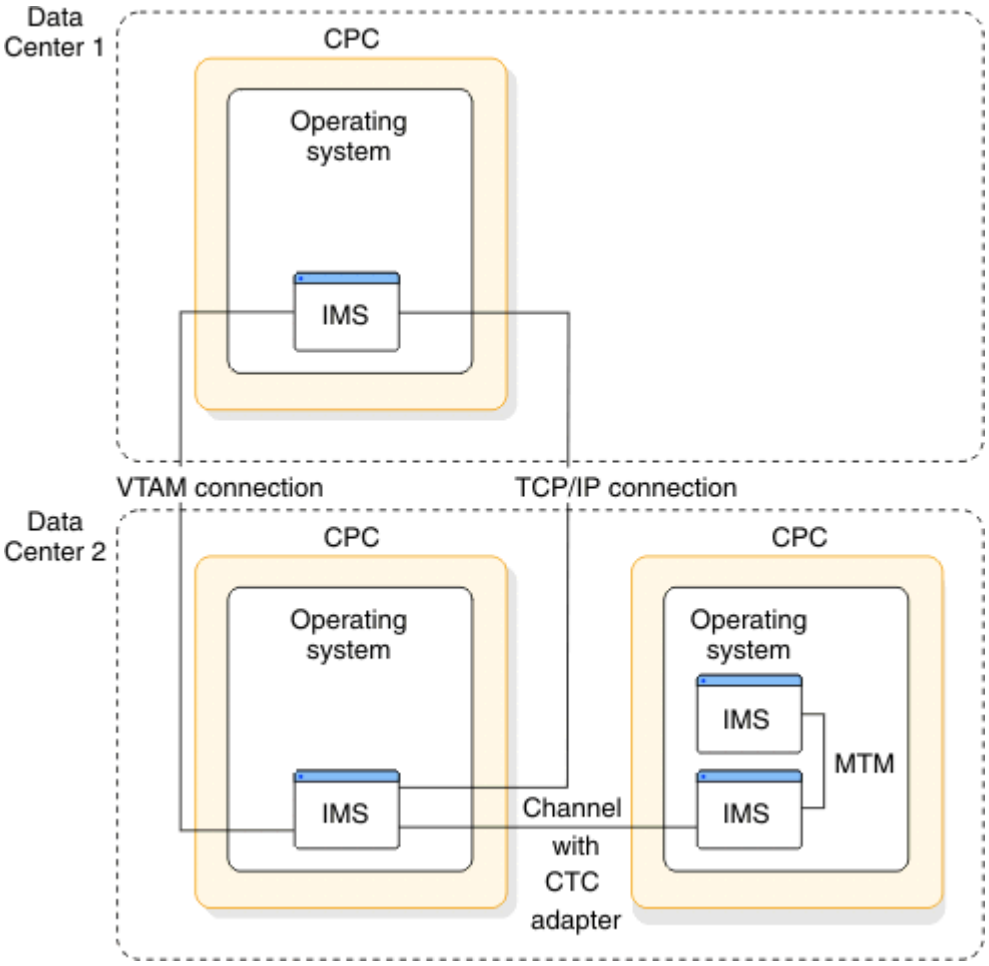


Figure 112. MSC physical link types

Related reference

CREATE MSPLINK command (Commands)
UPDATE MSPLINK command (Commands)
QUERY MSPLINK command (Commands)
MSPLINK macro (System Definition)

MSC logical links

A logical link relates a physical link to the transactions and terminals that can use that physical link.

Each IMS in an MSC network has one or more defined logical links.

A maximum of 1018 logical links are allowed for each IMS in the MSC network.

Two IMS systems that are defined to communicate with each other, each through a specific logical link, are called *partner systems*.

To establish connection between two IMS systems, each partner must have a logical-link definition. The two logical-link definitions must specify the same partner identifications and be assigned to the same physical link. The IMS system definition process assigns a number to each defined logical link. Logical link numbers are assigned sequentially, beginning with 1, in the order in which the links are defined. A logical link can be reassigned to a different physical link, but the two IMS systems must always communicate through a logical link partnership.

If you use the MSLINK stage-1 system definition macro, IMS also assigns a default name to each logical link, unless you specify a different logical link name in the label field on the MSLINK macro. Default logical

link names are DFSL*xxxx*, where *xxxx* is the logical link number. You can display or modify the logical link name by using the type-2 commands QUERY MSLINK and UPDATE MSLINK.

If you use the type-2 **CREATE MSLINK** command, you must specify the MSPLINK keyword to assign the logical link to a physical link. Otherwise, the logical link is not assigned to a physical link.

Type-1 commands require you to specify the logical link number to identify the target logical link. Type-2 commands require you to specify the logical link name to identify the target logical link.

To save logical links that are created or updated by using the type-2 commands across IMS cold starts, either export the logical link definitions to the IMSRSC repository or code the definitions to the MSC resources into stage-1 system definition macros.

If you use the IMSRSC repository to store dynamically defined MSC resources, ensure that automation and operational procedures that issue commands for MSC resources use type-2 commands, which specify link names, instead of type-1 commands, which specify link numbers. For example, instead of using the **/RSTART LINK 10** command to start a link, use the UPDATE MSLINK NAME(*logicallinkname*) START(COMM) command. During stage-1 system generation, the IMS system assigns numbers to logical links in the order in which the links are generated. However, the numbers for links are not stored in the IMSRSC repository. If logical links are referenced by using link numbers and are automatically imported from the IMSRSC repository, the numbers of the links are likely to change at the next IMS cold start.

With TCP/IP and VTAM physical link types, multiple logical links can use a physical link. When defining each physical link, you can specify how many logical links can share a physical link by using the SESSIONS keyword. The term *session* comes from VTAM and is generally synonymous with the term logical link. For VTAM link types, an active session is a logical link between partner systems.

The IMS system definition process does not require that a physical link be specified for each logical link. You can assign a physical link to the logical link online by using either the type-1 IMS command /MSASSIGN LINK or the type-2 IMS command UPDATE MSLINK NAME (linkname)SET(MSPLINK(msplinkname)). No communication between partners can occur until the assignment is made.

Related reference

CREATE MSLINK command (Commands)
UPDATE MSLINK command (Commands)
QUERY MSLINK command (Commands)
MSLINK macro (System Definition)

MSC logical link paths

Messages are routed in an MSC network by using *logical link paths*. Logical link paths identify the remote IMS system to which a message must be delivered and the local IMS system that is delivering the message.

You must define at least two corresponding logical link paths to submit a transaction at a terminal in an *input IMS system*, process the transaction in *destination IMS system*, and return a response to the terminal in the input IMS system: One logical link path in the input IMS system and another in the destination IMS system.

In the logical link path that is defined in the input IMS system, the input IMS system is defined as local and the destination IMS system is defined as remote.

In the logical link path that is defined in the destination IMS system, the destination IMS system is defined as local and the input IMS system is defined as remote.

In the definition of a logical link path, each IMS system is identified by one of its assigned system identifiers (SYSIDs), a numeric value from 1 to 2036.

IMS associates the SYSID of the input IMS system with the input transaction message and its corresponding response, so that the response can be returned to the input terminal.

Before it can be used, a logical link path must be assigned to a logical link. You can assign multiple logical link paths to a single logical link.

You can define a logical path either by issuing the type-2 command CREATE MSNAME or by coding the MSNAME stage-1 system definition macro. Because of the macro and command names, the term *MSNAME* is often used as a synonym for the term *logical link path*.

To save logical link paths that are created or updated by using type-2 commands across IMS cold starts, either export the modified logical link path definitions to the IMSRSC repository or code the changes to the MSC resources into stage-1 system definition macros.

Examples

Consider the MSNAME macro definitions of two logical link paths:

```
MSNAME  SYSID=(2,1)
MSNAME  SYSID=(3,1)
```

The first definition says that messages that use this logical link path are processed in the remote system whose local SYSID is 2. The second definition says that messages that use this logical link path are processed in the remote system whose local SYSID is 3. By using these definitions, the IMS system definition process assigns SYSID 1 to the IMS being defined and recognizes two remote systems with SYSIDs of 2 and 3. If a third path is defined with SYSID=(5,4), IMS would also assign SYSID 4 to the local system.

Transactions are assigned to logical link paths in the APPLCTN macro definition.

In the next example, consider the following application definitions, each with one transaction code defined:

```
APPLCTN     PSB=A
  TRANSACT  CODE=A
APPLCTN     PSB=B,SYSID=(2,1)
  TRANSACT  CODE=B
APPLCTN     PSB=C,SYSID=(3,1)
  TRANSACT  CODE=C
```

The SYSID keyword identifies the logical link path to be used for the transactions that are associated with the application. Transaction A is considered to be a local transaction, because the absence of the SYSID keyword indicates transaction A is only processed by the IMS being defined. Transactions B and C are remote transactions. Relating the application definitions to MSNAME definitions, IMS would return responses from transactions B and C to the IMS defined as SYSID 1, unless the application program specified an alternative destination for the response.

If messages that originate in the local system refer to any logical terminals in a remote system, the logical link definition must also include NAME macros to identify those remote logical terminals, unless directed routing is used.

Related reference

CREATE MSNAME command (Commands)

MSNAME macro (System Definition)

The MSC network and routing

The following topics explain the concepts necessary for MSC administration.

Remote and local systems

In an MSC network, a *local system* refers to a specific IMS where a message is entered. All other IMS systems are considered *remote systems* in regard to the specific local system.

A *local transaction* is a transaction that is processed in the same IMS in which it is entered. A *remote transaction* is a transaction that is entered into a IMS from a terminal or a link that is not processed in that IMS.

The following figure shows local and remote transactions and systems. When Transaction A is entered in IMS A, which is the local system, transaction A is processed locally. When Transaction A is entered on IMS B, which this time is the local system, it is sent across the MSC link to remote system IMS A and is processed remotely.

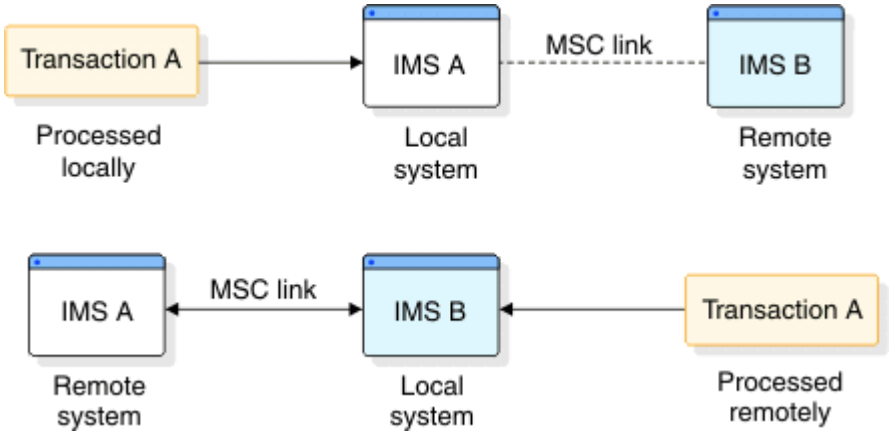


Figure 113. Remote and local transactions and systems

In the previous figure, transaction A, when entered from IMS A and processed in IMS A, is considered a local transaction. When transaction A is entered from IMS B and sent across an MSC link to be processed in IMS A, it is considered a remote transaction.

Flow of data within multiple systems

The flow of a transaction in an MSC network requires additional steps as compared to one IMS.

The general steps are illustrated in the figure below, and explained as follows:

- In the local system, a remote transaction entered from an LTERM is placed on the message queue of the local system with the destination of the remote transaction name **(1)**. (The message is queued to the MSNAME associated with the specified remote destination.)
- MSC removes the message from the message queue **(2)**, sends it across the MSC link **(3)**, and places it on the message queue of the remote system **(4)**.
- The remote system sends the message from the message queue to the application program to be processed **(5)**. After the application program processes the message, the program sends a reply.
- The remote system places the reply message, with a destination of the output LTERM, on its message queue **(6)**.
- MSC removes the message from the message queue of the remote system **(7)** and sends it back across the MSC link **(8)**.
- MSC places the message on the message queue of the local system **(9)** and sends it to the output LTERM **(10)**.

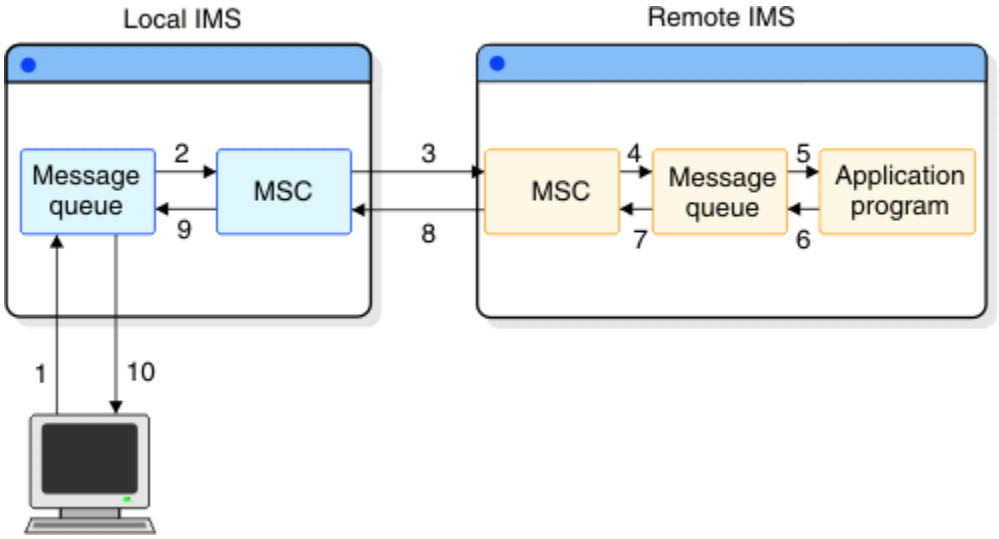


Figure 114. Remote transaction flow

Message routing

The message-routing function of MSC supports several types of message routing.

The types of message routing supported by MSC include:

- Routing of transaction messages from a terminal in one IMS to an application program in another IMS. The transaction can be defined as recoverable, nonrecoverable, response mode, or conversational.

  Restriction: Fast Path transactions across an MSC link are not supported. However, Fast Path transactions within the local system are supported.
- Routing of message switches from an LTERM in one IMS to an LTERM in another IMS and message switches between LTERMs in the same IMS. This support includes messages sent with a **/BROADCAST** command.
- Routing of response messages from an application program to the terminal that sends the transaction, or messages from an application program to an alternate terminal. Routing messages to alternate remote terminals (the transaction and the LTERM are in different IMS systems) requires that the alternate LTERM be defined as a remote LTERM. If directed routing or the TM and MSC Message Routing and Control User Exit routine is used, the alternate LTERM does not need to be defined as remote.
- Program-to-program switches between transactions in different IMS systems or within the same IMS.

Routing path

IMS passes messages from the local system to the remote system on a routing path. One or more IMS systems can be included in a routing path.

In the following figure, IMS B has a path to IMS A and back (path BA). Similarly, back-and-forth paths exist between IMS B and IMS D (path BD), IMS B and IMS C (path BC), and IMS A and IMS D (path AD). A path can go through an IMS in order to get to another IMS, such as path CAD between IMS C and IMS D. More than one path can exist between the same two IMS systems. IMS C and IMS D have two direct paths, CD1 and CD2, in addition to the indirect path CAD.

A total of 2036 paths are allowed in one MSC network.

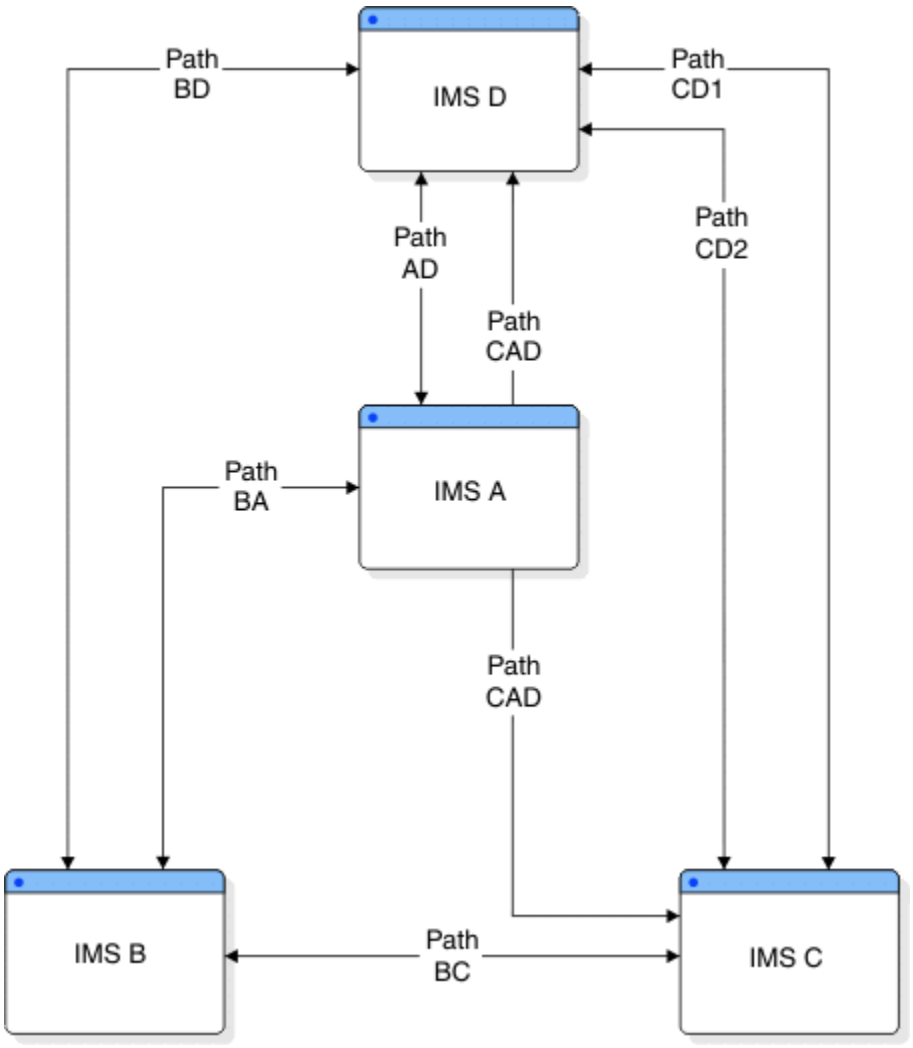


Figure 115. Routing path

Logical destinations

Message routing for an MSC network uses logical destinations, as does a single-system environment.

A destination is either an LTERM or a transaction code. A local destination resides in the local system, and a remote destination resides in a remote system. Within each local system, all local and remote destinations must be defined with unique names. In the figure below, all of the local and remote destinations are uniquely defined within each local system. Destinations that are defined remotely are not also defined locally within the same IMS. Similarly, destinations that are defined locally are not also defined remotely.

The same destination names can be used for local destinations in different IMS systems that are connected by MSC. The destination names cannot conflict with the global intent of the destination within the MSC network. For example, in the figure below, TRANAB is a local transaction in IMS B and IMS A. It is a remote transaction in IMS C. IMS C is referencing the local TRANAB in IMS A only and not the one in IMS B. IMS C cannot remotely reference TRANAB in IMS B. The destination system that is referenced in a remote destination is determined by the system identification (SYSID) value.

With directed routing and the TM and MSC Message Routing and Control User exit routine, you can send messages from an application program to a remote destination that is not explicitly defined in the destination system. The validation of the destination name in the transaction processing system is delayed until the message arrives in the local system. Delaying this validation provides more flexibility in the resource naming requirements.

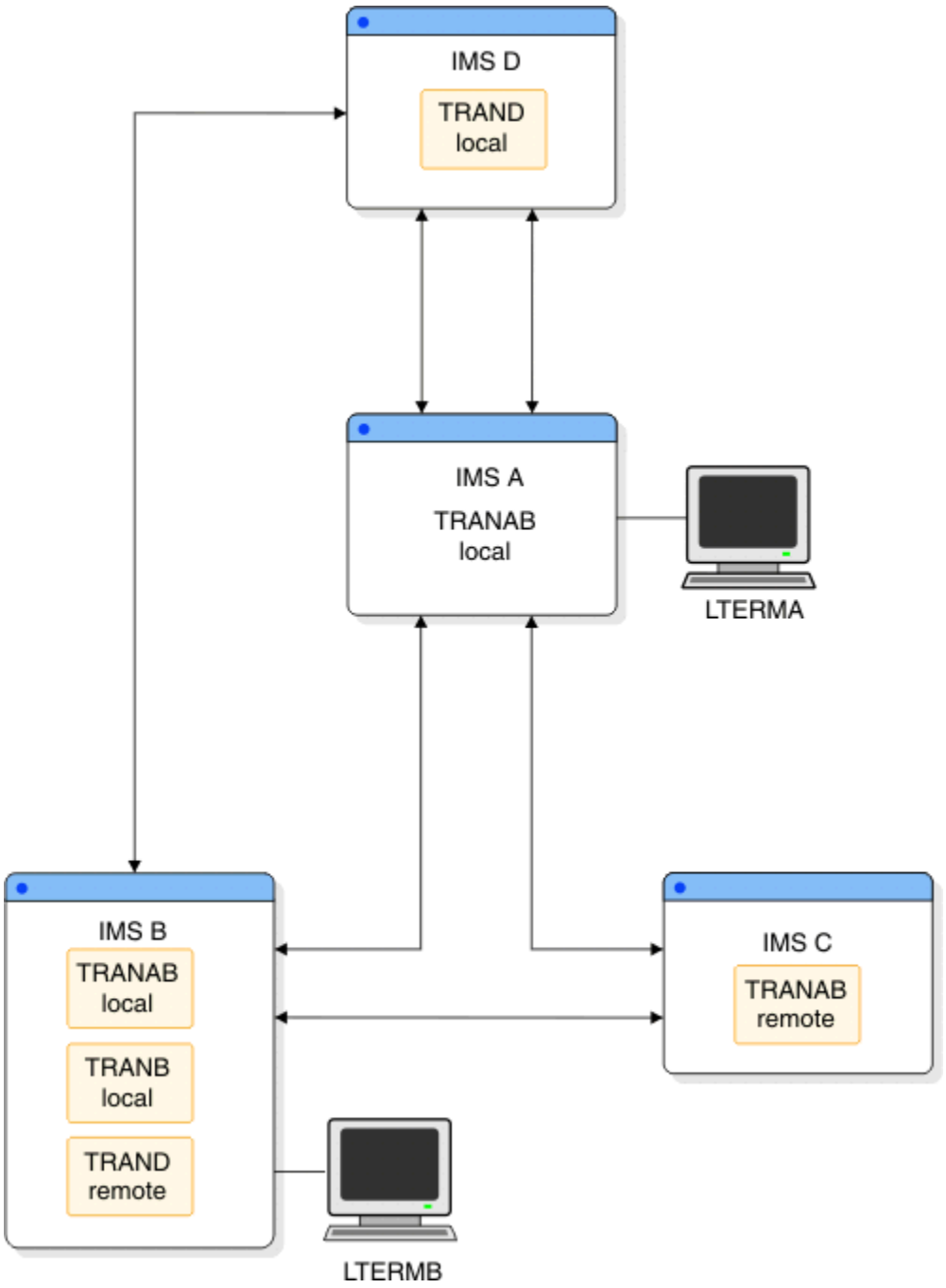


Figure 116. Logical destinations

Related concepts

"System identifiers (SYSIDs)" on page 677
MSC uses system identifiers (SYSIDs), two-byte numbers between 1 - 2036, to identify the IMS systems in an MSC network.

"Multiple Systems Coupling (MSC) directed routing" on page 683
MSC directed routing is a function of MSC that allows an application program to specify the IMS name (MSNAME) and destination within that IMS for a message to an LTERM or an application program.

Related reference

TM and MSC Message Routing and Control User exit routine (DFSMSCE0) (Exit Routines)

Input, destination, and intermediate systems

Depending on the message, an IMS can be either an input system, a destination system, or an intermediate system.

For example, in the following figure, the transaction message originates from LTERMB in IMS B and is routed over path BD to TRAND in IMS D. IMS B is the input system and IMS D is the destination system. This message is called a primary message. When the message is processed by the application program in IMS D and a reply is returned to the input terminal in IMS B, this reply message is called a response. For this response message, the destination and input system is IMS B. The destination of the message is LTERMB in IMS B. The input that caused this response message is also from LTERMB in IMS B, which is the primary transaction.

If a message is sent through one IMS and routed directly to another IMS for processing, the routing system is called the intermediate system. For example, if TRAND is sent from IMS B through IMS A to IMS D (path BAD), IMS A is the intermediate system and IMS D is the destination system. Similarly, IMS A and IMS D are intermediate systems for TRANC when TRANC is sent from IMS B through IMS D through IMS A to IMS C (path BDAC).

Remote destination names must be defined as remote in the input system and as local in the destination system. Intermediate systems, however, do not need to define the input, destination name, or the path back (a local SYSID) for the routed message. Only the path to the destination system needs to be defined. The path back might not even be through the intermediate system. Input and destination name checking is not performed on a message when it is routed through an intermediate system.

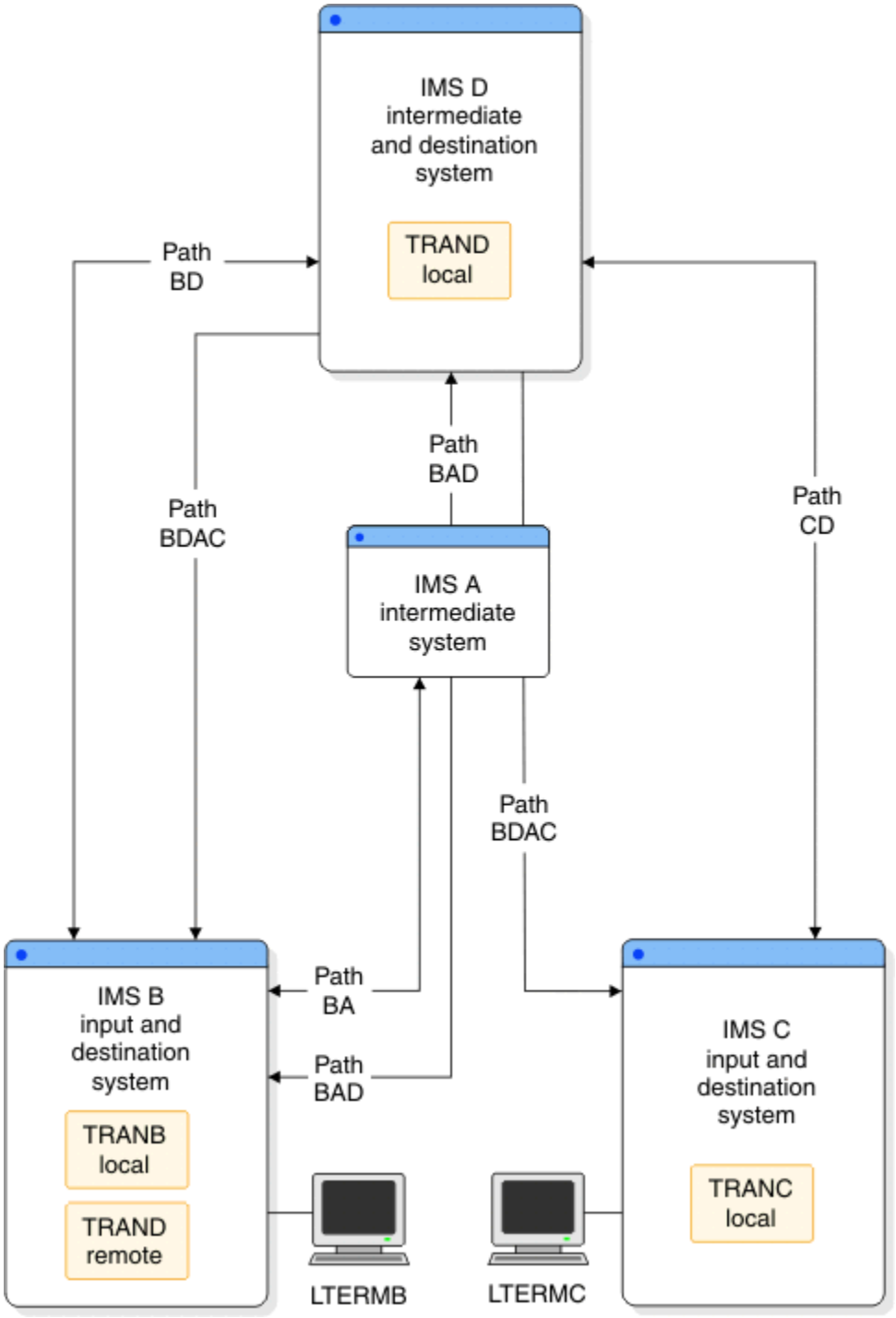


Figure 117. Input, destination, and intermediate systems

System identifiers (SYSIDs)

MSC uses system identifiers (SYSIDs), two-byte numbers between 1 - 2036, to identify the IMS systems in an MSC network.

Each IMS system in an MSC network is assigned one or more SYSIDs. The SYSIDs are *local* to the IMS system that they are assigned to and *remote* to the other IMS systems in the MSC network.

The local SYSIDs must be unique within an MSC network. If an MSC network connects to an IMSplex with shared queues, the IMS systems in the shared queues group can share local SYSIDs.

A local SYSID and a remote SYSID are paired to define a *logical link path*. A logical link path is assigned to a logical link between the two IMS systems that the local and remote SYSIDs represent. The term *MSNAME* is often used as a synonym for the term logical link path.

As messages travel through an MSC network, the local SYSID identifies the point of origin of the message and the remote SYSIDs identify the destination of the message. Generally, the IMS system identified by the remote SYSID is where a transaction message is processed and the local SYSID is where the response to the transaction must be returned. However, a remote SYSID can also represent an intermediate IMS system that doesn't process the transaction, but instead passes it to another IMS system in the network.

Local SYSIDs can be assigned to an IMS system in the following ways:

- MSNAME stage-1 system definition macro.
- MSC section of the DFSDFxxx member of the IMS PROCLIB data set.
- IMS type-2 command **CREATE MSNAME**.

You can change SYSIDs online by using the IMS type-2 command **UPDATE MSNAME** NAME(*msname*) SET(SIDL|SIDR).

In the following figure, IMS B has local SYSIDs 1, 2 and 3. IMS A has local SYSID 4. IMS C has local SYSID 5. IMS D has local SYSIDs 6 and 7. IMS B has local SYSIDs 1, 2, and 3 because of the three MSNAME macros with local SYSIDs of 1, 2, and 3.

In the figure, IMS B has three paths to remote SYSIDs 6 (IMS D), 7 (IMS D), and 5 (IMS C). IMS D and IMS B cannot route messages to IMS A, because they do not have paths to SYSID 4 (IMS A). IMS A does, however, have a path to SYSID 7 (IMS D) and SYSID 2 (IMS B). IMS A cannot send messages that originate in IMS A to SYSID 7 or SYSID 2, because the source SYSID (SYSID 4) is not recognized by IMS B or IMS D. In this configuration, IMS A can only function as an intermediate system for IMS B and IMS D. Another path (MSNAME) must be defined for IMS B to communicate with IMS A and IMS D.

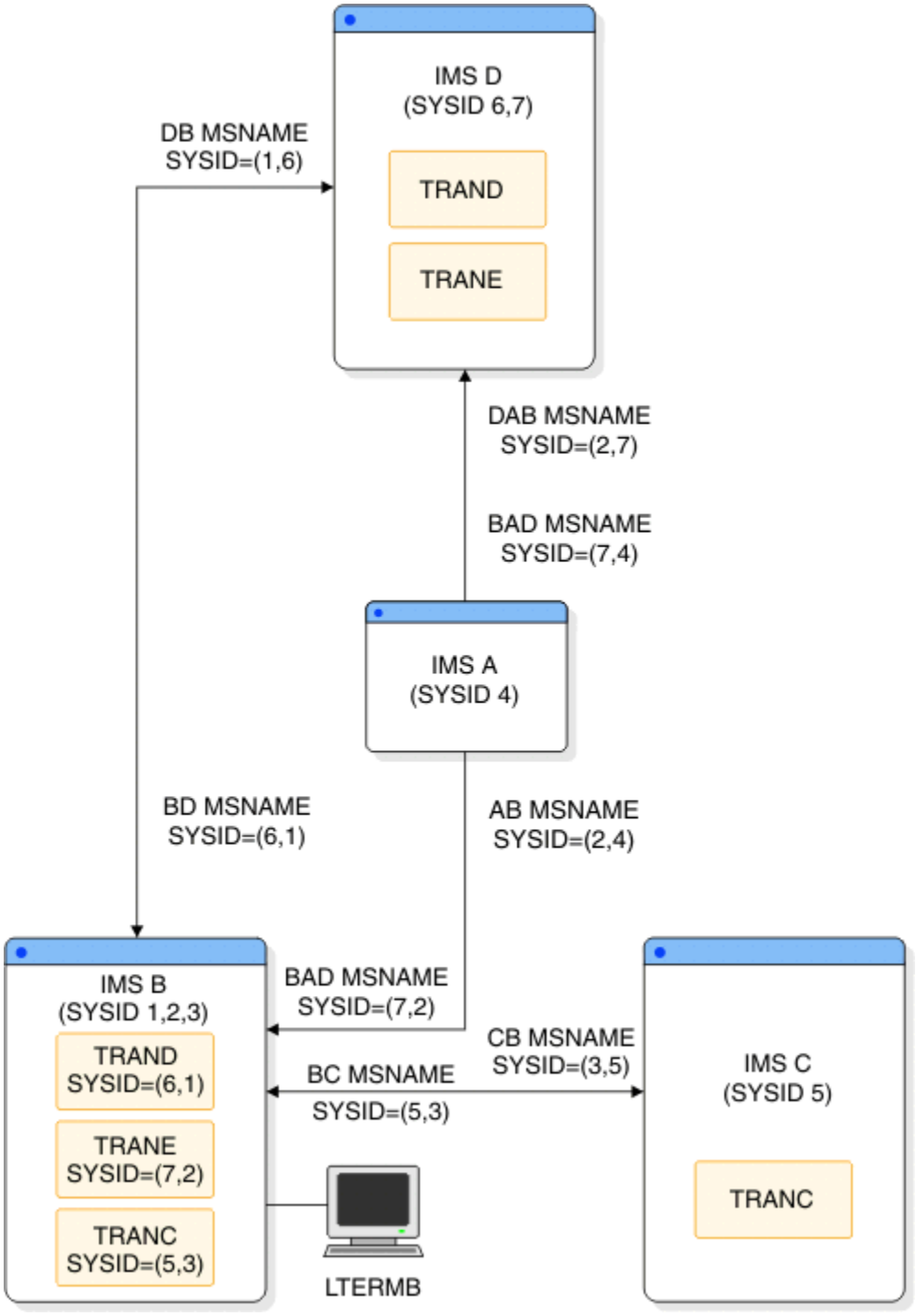


Figure 118. System identifiers (SYSIDs)

Related concepts

"MSC logical link paths" on page 670

Messages are routed in an MSC network by using *logical link paths*. Logical link paths identify the remote IMS system to which a message must be delivered and the local IMS system that is delivering the message.

Related tasks

"Defining a SYSID" on page 693

System identifiers (SYSIDs) are 2-byte numbers between 1 - 2036 that identify the IMS systems in an MSC network.

Related reference

MSNAME macro (System Definition)

MSC section of the DFSDFxxx member (System Definition)

CREATE MSNAME command (Commands)

Routing messages with the destination name and SYSIDs

Messages in an MSC network contain information that makes it possible to route the message between IMS systems.

When a remote transaction is entered from an LTERM, a transaction message is built and queued on the message queue. This message contains information that is needed to route the message to its remote destination:

- Remote destination name (transaction code)
- Local or source LTERM name of the LTERM that entered the transaction
- Remote SYSID value
- Local or source SYSID value

If the transaction is processed by the application program and secondary messages are sent to other transactions (program-to-program switches), these messages have the destination name and SYSID of the switched-to transaction. The source (origin) name and SYSID remain the same. That is, the source SYSID and name never change. This facilitates sending the response message back to the input LTERM, regardless of where the transaction is processed.

Any IMS that locally processes a message that is received from a remote system must have a local SYSID defined to it that is the same as the remote SYSID of the message. It must have a path back to the source (origin) of the message. In the following figure, if TRANA is entered from LTERMB in IMS B, it is sent across the path MSNAME=(4,2) to IMS A. The destination name and SYSID of the message are TRANA and 4. The source name and SYSID are LTERMB and 2. IMS A accepts the message and processes it. It has SYSID=4 defined as local and has a path back to IMS B with the destination SYSID=2. If TRANA in IMS A issues a program-to-program switch to TRAND in IMS D, the destination name and SYSID are TRAND and 5. The source name and SYSID remain LTERMB and 2. IMS D accepts and processes the transaction. The transaction has SYSID=5 defined locally and has a path back to IMS B with destination SYSID=2.

If TRAND in IMS D sends a response back to input LTERMB, the response message has a destination name and SYSID of LTERMB and 2, and the source name and SYSID are also LTERMB and 2.

The input LTERM, LTERMB, is never defined as remote in IMS A or IMS D, yet the response message is returned to LTERMB in IMS B. IMS carries the originating LTERM name and SYSID in the primary message and all secondary messages. IMS knows where to route the response message when the application program responds to the input LTERM.

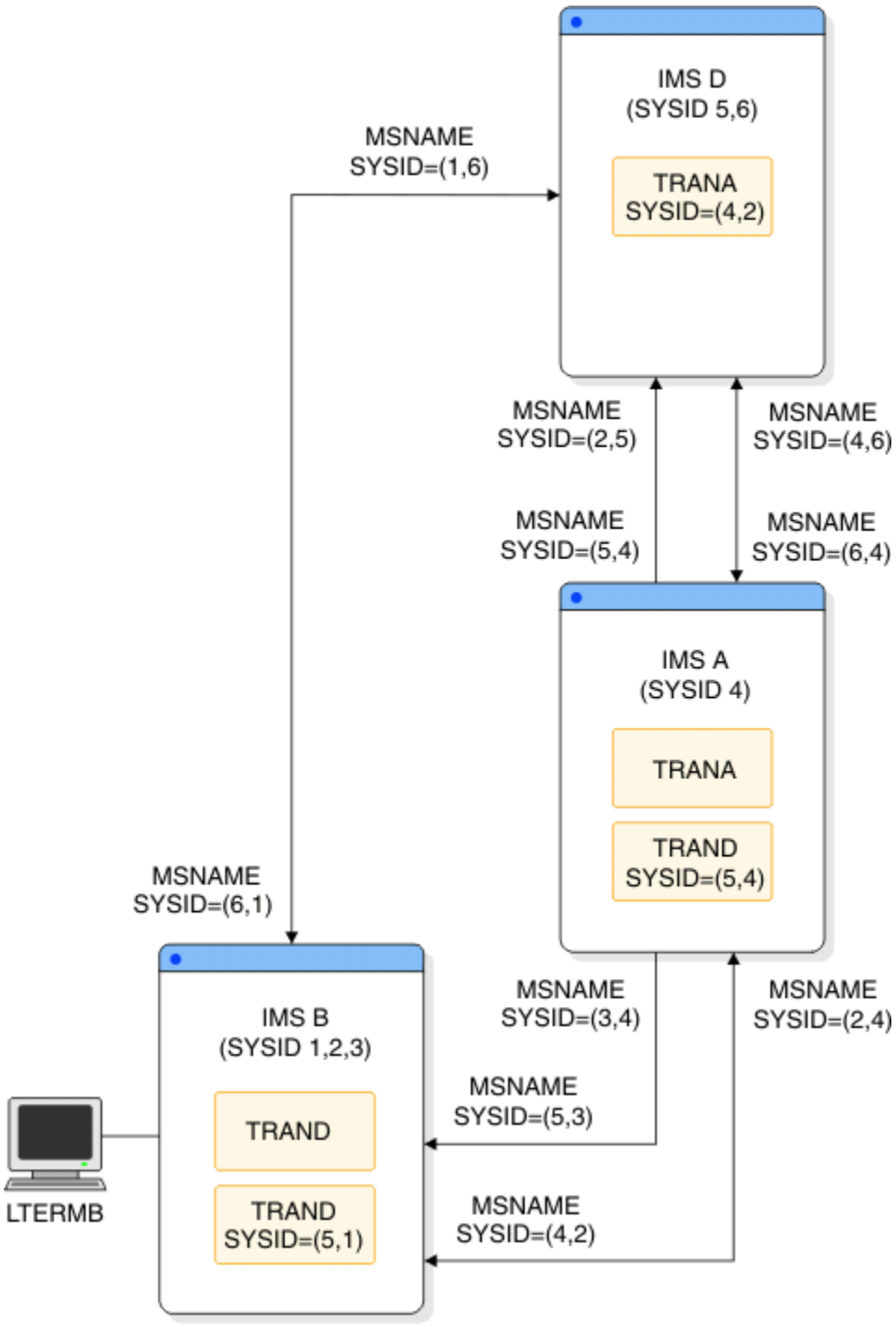


Figure 119. Message routing

Remote LTERMs

A *remote LTERM* is a logical terminal that does not reside on the local system.

Recommendation: In remote IMS systems, define the input LTERM from the local IMS system as a remote LTERM only to send message switches, remote broadcasts, or messages from an application program to an alternate remote terminal (alternate PCB). In other cases, IMS remembers the input LTERM and can return response messages from remote systems to the input LTERM without the input LTERM being defined in the remote system.

You can define remote LTERMs by using ETO MSC descriptors. The ETO MSC descriptor relates remote LTERMs to statically defined MSC links.

You can also define remote LTERMs dynamically by using the type-2 CREATE LTERM command.

To help return response messages to the input (source or origin) LTERM, IMS carries the source LTERM name and SYSID in the remote message. To send a response message to an LTERM other than the source LTERM, you must define a remote LTERM.

- If you use the NAME macro, in the remote system, define NAME macros that have the name of the local terminal in the local system. Associate the NAME macro with the MSNAME that defines the destination SYSID of the local system.
- If you use the CREATE LTERM command to dynamically create the remote LTERM in the remote system, define the remote LTERM name with the same name as the local terminal in the local system. Associate the remote LTERM with the logical link path (MSNAME) that defines the destination SYSID of the local system.

For example, in the following figure, IMS A has local LTERMA, IMS B has local LTERMB, and IMS D has local LTERMD. IMS B can send message switches and remote broadcasts from LTERMB to LTERMD because LTERMD is defined remotely in IMS B. LTERMD is not actually defined with SYSID=(5,2), but it assumes those SYSIDs when the message is issued. LTERMD is associated with MSNAME BAD, which is defined with SYSID=(5,2) in IMS B. This association is established by placing the NAME macro for LTERMD after the following MSNAME macro:

```
BAD MSNAME SYSID=(5,2)
        NAME LTERMD
```

This naming differs from remote transaction definitions, which are explicitly defined with remote and local SYSIDs.

Also, LTERMA in IMS A can send message switches or remote broadcasts to LTERMD in IMS D. Similarly, LTERMD in IMS D can send message switches or remote broadcasts to LTERMA in IMS A. Both IMS A and IMS D have remote LTERM specifications for the other system's local LTERM.

TRAND in IMS D can send alternate messages to LTERMB in IMS B or to LTERMA in IMS A. MSNAMEs are defined with names for the purpose of:

- Displaying the queue counts for the named logical link path
- Stopping the sending of all messages from a terminal except those continuing a conversation
- Starting the logical link path
- Purging the logical link path in an MSC network for which input is stopped
- Allowing application programs to use directed routing
- Reassigning them to different links by using either the type-1 command `/MSASSIGN MSNAME msname LINK link#` or the type-2 command `UPDATE MSNAME NAME(msname) SET(MSLINK(linkname))`

In the following figure, the name of each MSNAME is the two-character name preceding MSNAME.

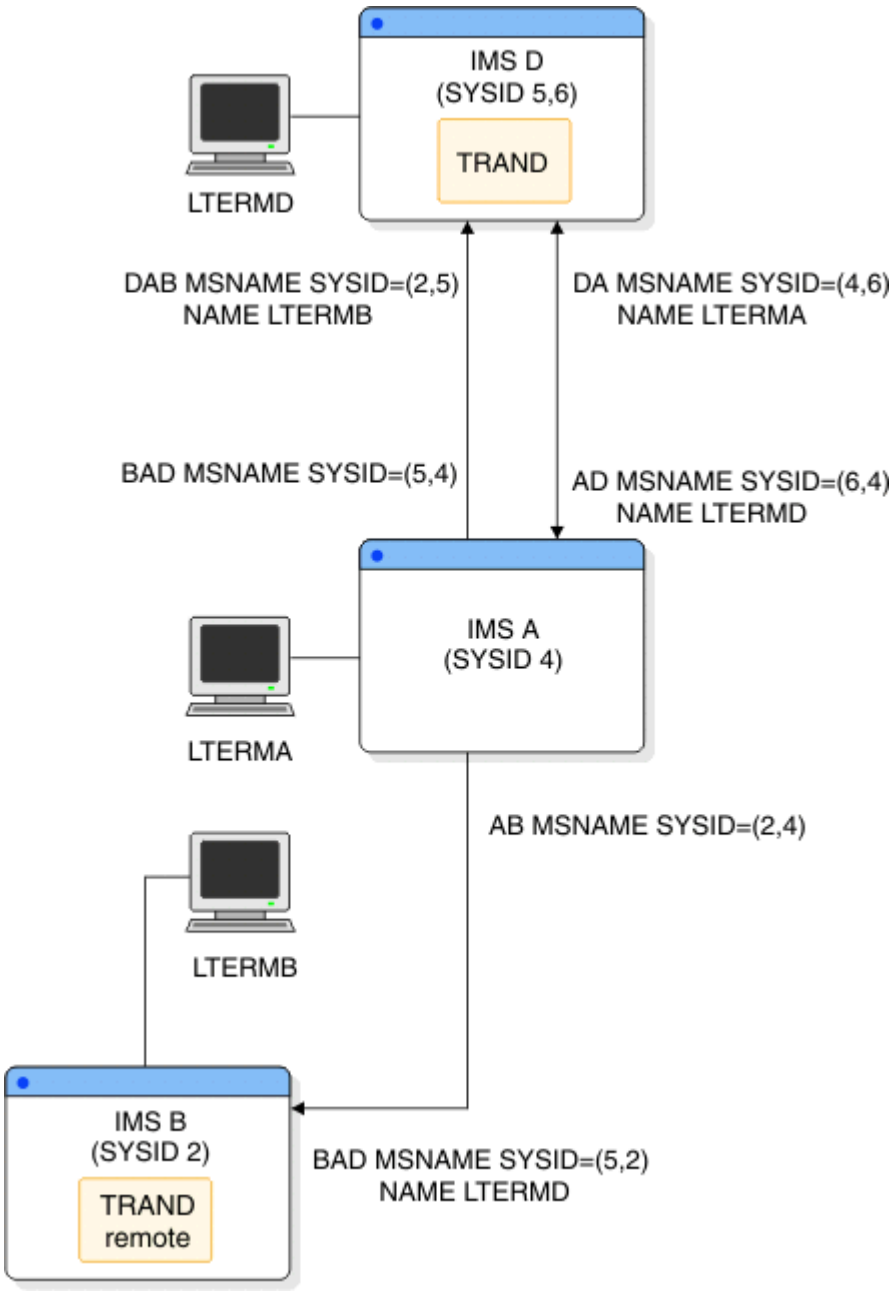


Figure 120. Remote LTERMs

Related reference

MSC descriptor format and parameters (System Definition)

CREATE LTERM command (Commands)

Multiple Systems Coupling (MSC) directed routing

MSC directed routing is a function of MSC that allows an application program to specify the IMS name (MSNAME) and destination within that IMS for a message to an LTERM or an application program.

The receiving application program can determine the MSNAME of the IMS that originally scheduled it. With directed routing, the specified remote destination (a transaction or an LTERM) in another IMS does not need to be declared explicitly in the IMS system definition for the sending IMS. These logical (local) names for terminals enable different IMS systems in the MSC network to use the same logical names for terminals and transaction codes. Names must still be unique within a given IMS. The Multiple Systems Verification utility (DFSUMSV0) cannot detect errors associated with MSC directed routing.

Restrictions:

- MSC directed routing does not support a program-to-program switch between conversational transactions.
- MSC directed routing does not support a program-to-program switch from a nonconversational transaction to a conversational transaction. For example, a conversational transaction in System A cannot use directed routing to perform a program-to-program switch to invoke a conversational transaction in System B.
- Response mode cannot be propagated on a DL/I ISRT call in a directed routing transaction.

Related concepts

Communicating with other IMS TM systems using Multiple Systems Coupling (Application Programming)

Remote destination verification

To maintain system integrity and prevent errors, an IMS in an MSC network verifies all specified destinations, unless MSC directed routing is used.

When MSC directed routing is used, IMS only ensures that a program-to-program switch is not being performed from a nonconversational transaction to a conversational transaction. Remote destination verification occurs when a message is received from a terminal or on receipt of an application program reply if a remote destination is specified for the message. Destination verification occurs as follows:

Destination
Verified For:

LTERM
Destination type: The original destination must have been a logical terminal.

Transaction
Destination type: The original destination must have been a transaction.

Transaction attributes: The following attributes must be consistent in the transaction definitions in the input and destination systems:

- Single segment or multiple segment
- Recoverable or irrecoverable
- Conversational or nonconversational
- Inquiry or update

The SPA size for a conversation from a remote system cannot be changed except by the remote system on an insert.

When an invalid destination is recognized, IMS cancels the message, sends an error message to the input terminal and to the master terminal of the local system, and logs an invalid request. If the message is conversational, the Conversation Abnormal Termination exit routine (DFSCONE0) is called in the input system, and the conversation is terminated.

Chapter 39. Administering Multiple Systems Coupling

The following topics describe the system administration activities required when you connect two or more IMS online systems in a network using MSC.

LU 6.2 application programs can process transactions on remote IMS systems. ETO also supports remote LTERMs.

Related concepts

"Remote LTERMs" on page 681
A *remote LTERM* is a logical terminal that does not reside on the local system.

Design considerations for multiple systems

The major design goals for MSC are to minimize resource consumption by defining suitable connections between the systems, balance resource demand by distributing functions among systems to obtain acceptable performance, and to program design considerations for multisystem conversational transactions.

The design and tuning recommendations that apply to a single IMS system are applicable to each IMS system in an MSC environment. Resource demand and consumption are taken into account when defining systems that are part of an MSC configuration.

An IMS transaction that is processed in a local system uses the same hardware and software resources that it would in a non-MSC environment. Transactions that are processed in a remote system require additional resources. In addition to resources used to transmit the transaction over physical links to the remote processor and to receive the response from the remote processor, resources are needed for message queuing and logging. Performance considerations are directly related to minimizing the resources consumed by remote processing and balancing the resource demand between several processors in an MSC configuration.

Minimizing resource consumption

You can minimize resource consumption in a number of ways when you design an MSC system.

To minimize resource consumption, do each of the following:

- Design the environment so that as many transactions as possible are processed locally.
- Provide physical links that go directly from local to remote systems; no intermediate systems should be involved in the transaction routing process. Transactions that must be routed through intermediate systems require additional processor activity, message queue activity, and logging activity.
- Design the message queue buffer pool in each processor to eliminate unnecessary message queue I/O activity.
- Design the queue buffer size to be large enough to hold the transaction input message and output response message in a single queue buffer.
- Define the physical link buffer sizes large enough to hold the message prefix plus all the segments of most messages. The physical link buffer size defines the default buffer size for all logical links assigned to a physical link. You can specify different buffer sizes for each logical link by using the **UPDATE MSLINK** NAME(*linkname*) SET(BUFSIZE(*new_bufsize*)); however, the specifications for the buffer sizes for any given logical link must be the same in both of the IMS systems that the logical link connects.

© Copyright IBM Corp. 1974, 2020

Controlling the bandwidth of MSC links

You can control the bandwidth of CTC, MTM, and VTAM MSC link types by increasing MSC link buffer sizes and turning MSC bandwidth mode on and off.

TCP/IP physical link types always run in bandwidth mode. Bandwidth mode cannot be turned off for TCP/IP links.

When bandwidth mode is off, MSC sends a maximum of one message or response per I/O send or write operation. When bandwidth mode is on, IMS consolidates the following messages and responses in the same buffer:

- All messages that are queued and ready to be sent
- Any responses that are owed for messages received

Increasing the link buffer size allows more messages and responses to be sent simultaneously. Valid MSC link buffer sizes are 1024 bytes to 65536 bytes. The physical link defines the default buffer size for logical links associated with it.

You can set the default value by specifying the BUFSIZE keyword on the MSPLINK macro during system definition, or you can set it dynamically by specifying the BUFSIZE keyword on the CREATE MSPLINK command. You can then modify the buffer sizes for each logical link in an online IMS system by using the UPDATE MSLINK or the UPDATE MSPLINK command; however for any one logical link you must specify the same buffer size in both IMS systems at each end of the logical link.

To change bandwidth mode and link buffer sizes, use either the type-2 command UPDATE MSLINK or the type-1 command /UPDATE MSLINK. To display the status of bandwidth mode and the size of the buffers, use the type-2 command QUERY MSLINK or the type-1 command /DISPLAY LINK OPTION BUFSIZE.

You cannot set bandwidth mode during system definition. It must be set with a command. After you establish a bandwidth mode and buffer size, the mode and size selected remain in effect across link restarts and IMS warm starts. Bandwidth mode is set off and the buffer size is restored to the SYSDEF value across an IMS cold start.

Related reference

MSPLINK macro (System Definition)
CREATE MSPLINK command (Commands)
UPDATE MSLINK command (Commands)
UPDATE MSPLINK command (Commands)
QUERY MSLINK command (Commands)

Balancing resource demand

In an MSC environment with two or more processors, try to distribute the workload in a way that avoids excessive use of any one processor.

You can distribute the workload by distributing IMS applications and their associated transactions and terminals between the available processors. Depending on the complexity of the application and the capability of the processor, you can avoid overloading any single processor.

If the current design of the databases is such that the databases and their associated applications cannot be distributed across the available processors, you can:

- Duplicate inquiry-only databases; this allows more than one system to reference the whole of the database. (This is called vertical partitioning.)
- Split the databases into several component databases. (This is called horizontal partitioning.) The component databases must be completely independent for distribution among the available processors. For example, it might be possible to divide a database by key range intervals. The new databases and their associated applications can then be distributed among the existing IMS systems, and you can use the Terminal Routing exit routine to route incoming transactions to the correct IMS system. Another possibility is to divide the database by geographic area. Each IMS system could

process the transactions that refer to the databases for its own geographic area and route transactions that refer to a remote geographic area.

In addition to balancing the workload across processors, you might also need to balance the workload on physical links. This occurs when a physical link between two systems is of the SDLC type and multiple physical links have been installed. You can balance the workload on physical links by:

- Specifying, during IMS system definition, proper logical link paths and logical links for each remote application.
- Using a user-written TM and MSC Message Routing and Control user exit routine (DFSMSCE0) to distribute the transaction load on each of the alternative physical links.

Planning for conversational processing

Conversational processing is available to terminals that are attached to any IMS in an MSC network to the same extent as if they were in a single-system environment.

The following differences apply to conversational processing in an MSC network:

- All transactions used in a conversation must be defined as conversational in each IMS of the MSC network.
- The input system controls the conversational resources for the duration of the conversation. When the input system receives a conversational transaction, it inserts the scratchpad area (SPA) as the first message segment, and routes the message to the destination application program.
- Any system in the MSC network can process any step in the conversation.
- Program-to-program switches can be routed from system to system.
- SPAs that are used in multisystem conversations must follow these rules:
  - For the SPA ISRT, conversational program-to-program switches can occur to a transaction with a SPA that is equal, larger, or smaller in size.
  - The minimum size of a SPA is 16 bytes (X'10'), and the maximum size is 32767 bytes (X'7FFF').

Generally, terminal operators and application programs are unaware of whether a conversation is multisystem.

Exceptions:

- Suppose a conversational program inserts a message to a response alternate PCB in a remote system. By implication, this destination is in the input system and is verified by the input system. Destination verification in this case involves confirming that the specified logical terminal is still assigned to the input terminal. If the logical terminal has been reassigned, the input system activates the Conversation Abnormal Termination exit routine, and terminates the conversation. The status code that is returned to the application program is blank, indicating success. This status code is blank even when the actual result is unsuccessful due to the condition described.
- Suppose an application program executing in a system other than the input system uses the SPA to specify a transaction code, thereby passing conversation control to another program. If the specified transaction code is invalid, the input system activates the Conversation Abnormal Termination exit routine, and terminates the conversation. No status code is returned to the application program.

MSC support for APPC/IMS and OTMA includes the following IMS transaction types:

Conversational
Nonconversational
Response mode
Non-response mode

Restriction: MSC does not support Fast Path.

Routing exit routines with conversations

You can use the program routing and link routing entry points of the TM and MSC Message Routing and Control User exit routine (DFSMSCE0) to input messages that start a conversation.

It is not applicable at any other conversational step, because the application program, not the input terminal, provides the destination for continuation of the conversation.

Remote destination verification for conversations

Destinations for program-to-program switches are verified in the system in which the program requesting the switch executes, except where MSC directed routing is used.

If MSC directed routing is not used and the destination is valid, the system sends the SPA and the message to the destination transaction. If the destination is invalid, the system returns a status code to the application program.

If MSC directed routing is used, IMS ensures only that a program-to-program switch is not being performed from a nonconversational transaction to a conversational transaction. If the destination is valid, the system sends the SPA and the message to the destination transaction. If the destination is invalid, the system does not route the SPA and the conversation is terminated.

Destination verification for a message to the input terminal is performed by the input system. The specified logical terminal must still be assigned to the input terminal. The input system also verifies, except when MSC directed routing is used, the next transaction that is specified in the SPA. If the destination is invalid, the input system invokes the Conversation Abnormal Termination exit routine and terminates the conversation. No status code is returned to the application program.

Saving truncated data in the SPA

The SPA=STRUNC option on the TRANSACT macro applies to conversations that use program switches to other transactions that have different sized SPAs.

- If you use the SPA=STRUNC option, IMS preserves all of the data in the SPA, even when a program switch is made to a transaction that is defined with a smaller SPA. The transaction with the smaller SPA does not see the truncated data. However, when this transaction switches to a transaction with a larger SPA, the truncated data is used. IMS tracks the longest data that is inserted to the SPA to determine the truncated data length.

  Example: If you have the three transactions:

  ```
  TRANA SPA=100
  TRANB SPA=50
  TRANC SPA=150
  ```

  If the application program for TRANA switches to TRANB, the last 50 bytes of the SPA for TRANA are not sent to TRANB. If the application program for TRANB subsequently switches to TRANC, the SPA that is received by TRANC contains the following 150 bytes:

  – The first 50 bytes from the SPA that was inserted by TRANB
  – The second 50 bytes from the second 50 bytes that was inserted by TRANA
  – The third 50 bytes are binary zeros.
- If you do not use the SPA=STRUNC option, the truncated data is lost. In the previous example, the second 50 bytes that is received by TRANC would be binary zeros.

Restriction: Truncated SPA data from a previous transaction is lost if it is sent using MSC to an IMS 5.1 (or earlier) system.

Conversation termination

A conversation can be terminated by either the application program or the terminal operator.

An application program terminates a conversation by inserting a message to the input terminal with a SPA that contains either blanks as the transaction code or the transaction code of a nonconversational transaction.

A conversational transaction can also be terminated by the **/EXIT** command.

An IMS shutdown does not terminate conversations. The conversation is continued after IMS restarts, unless an IMS cold start occurs or the conversation is terminated by an operator command such as **/EXIT**. If the input system is shut down and subsequently cold starts, all the conversations that it controls are lost. If using Resource Manager (RM) and the status recovery mode is GLOBAL, an IMS conversation can survive an IMS cold start because the status is stored in RM. The input system cancels any conversational messages it receives for input terminals that were previously involved in active or held conversations. IMS retains conversations over a restart. An IMS cold start is an unusual procedure, and can cause many problems beyond losing conversation state.

If a remote system is shut down when a conversational step is processing or in its queue, and it is subsequently cold started, all references to the conversation are lost. A conversation that is lost in this way must be specifically canceled in the input system by the **/EXIT** command.

Abnormal conversation termination

Several different events can lead to the abnormal termination of a conversation.

A conversation is abnormally terminated if any one of the following events occur:

- The conversational application program abnormally terminates.
- An invalid destination is recognized in the input system or in the remote system (for a conversation response, a program-to-program switch, or in the SPA).
- A conversational message is inserted into a terminal whose conversation was terminated.
- Destination verification fails for a conversational message.
- No output was generated in the application program.

The conversation's SPA (along with an indication of the cause of termination) is passed to the Conversation Abnormal Termination exit routine in the input system.

Defining Multiple Systems Coupling resources

Multiple Systems Coupling (MSC) resources, such as physical links, logical links, and system identifiers, can be defined either dynamically while IMS is running (using IMS type-2 commands) or statically during IMS system definition (using macros).

Before you can define MSC resources dynamically, both MSC and dynamic definition for MSC must be enabled in the IMS system. MSC is enabled either by specifying the MSC= execution parameter in the IMS startup procedure or by defining at least one MSC link with stage-1 system definition macros during IMS system definition. Dynamic definition for MSC is enabled by specifying MSCRSCS=DYN in the MSC section of the DFSDFxxx PROCLIB member.

After MSC and dynamic definition is enabled, you can create, update, and delete MSC resources by using IMS type-2 commands, such as CREATE MSPLINK, UPDATE MSPLINK, and DELETE MSPLINK. To save changes to MSC resources that are made dynamically across a cold start, either export the definitions to the IMSRSC repository or code the changes to the MSC resources into stage-1 system definition macros. To delete resources from the IMSRSC repository, issue the **DELETE DEFN** command. Otherwise, changes to MSC resources that are made dynamically are saved across warm and emergency restarts only. MSC does not support resource definition data sets (RDDSs).

If you define MSC resources statically during IMS system definition, you do not need to specify the MSC startup parameter, although doing so can avoid confusion. To enable MSC in an IMS online system by

using static definitions only, your IMS system definition must include three macros: MSPLINK, MSLINK, and MSNAME.

You also need to define transaction codes for each IMS system that has any part in transaction entry or processing. You can define transaction codes dynamically or by using system definition macros. An individual system can play several roles: it can be the input system, it can be an intermediate system responsible for routing transactions, or it can be a destination system in which the transaction is processed.

Using MFS is the same in an MSC network as in a single-system environment. If a message is created in one IMS for a terminal that is attached to another IMS, the required message and format descriptions must be available to the IMS to which the terminal is attached, and definitions with the same name must be defined identically in each IMS.

Enabling MSC in an IMS system

Multiple Systems Coupling can be enabled in an IMS system either by coding the MSC=Y execution parameter in your startup procedure or, unless MSC=N is specified, by defining at least one MSC link during IMS system definition.

Recommendation: To ease operations when dynamic MSC is enabled and MSC resources are no longer defined using the system definition process, specify the MSC=Y execution parameter, even if MSC resources are defined with the system definition process.

Enabling DRD for MSC

Enable dynamic definition for MSC resources by specifying MSCRSCS=DYN in the MSC section of the DFSDFxxx member of the IMS PROCLIB data set.

Before you can enable dynamic resource definition for MSC resources, the Common Service Layer (CSL) must be enabled with at least the Structured Call Interface (SCI) and the Operations Manager (OM).

After you enable dynamic resource definition for MSC resources, you can create, modify, and delete MSC resources in an online IMS system by using IMS type-2 CREATE, UPDATE, and DELETE commands.

Any changes that are made to MSC resources dynamically are not saved across a cold start unless you code the changes to the MSC resources into stage-1 system definition macros or export the changes to the IMSRSC repository before the cold start occurs.

To enable dynamic definition, perform the following steps:

1. Enable MSC by specifying MSC=Y in the DFSPBxxx member of the IMS PROCLIB data set, or by defining at least one MSC link during IMS system definition.
2. Specify MSCRSCS=DYN in the MSC section of the DFSDFxxx member of the IMS PROCLIB data set.
3. Cold start IMS.
4. Confirm that dynamic definition is enabled for MSC resources by issuing the QUERY MEMBER TYPE(IMS) command.

   When dynamic resource definition is enabled for MSC resources, the command output includes DYNMSC in the local attributes.

Related tasks

“Enabling the IMSRSC repository for MSC resources” on page 691
You can enable the IMSRSC repository to store all MSC resource definitions in a single, centralized location for all members of an IMSplex. If you enable the IMSRSC repository for MSC resources, the MSC resources that are created and updated dynamically can be saved across an IMS cold start.

“Enabling MSC in an IMS system” on page 690
Multiple Systems Coupling can be enabled in an IMS system either by coding the MSC=Y execution parameter in your startup procedure or, unless MSC=N is specified, by defining at least one MSC link during IMS system definition.

Related reference

DFSDFxxx member of the IMS PROCLIB data set (System Definition)

MSC section of the DFSDFxxx member (System Definition)

Enabling the IMSRSC repository for MSC resources

You can enable the IMSRSC repository to store all MSC resource definitions in a single, centralized location for all members of an IMSplex. If you enable the IMSRSC repository for MSC resources, the MSC resources that are created and updated dynamically can be saved across an IMS cold start.

Before you enable the IMSRSC repository for MSC resource definitions, ensure that the IMSRSC repository is defined and enabled.

1. To enable the IMSRSC repository for dynamically defined MSC resources, specify MSCREPO=Y in the MSC section of the DFSDFxxx proclib member.
2. Cold start IMS.

After you enable the IMSRSC repository for dynamically defined MSC resources, the online resources can be automatically exported to the IMSRSC repository at IMS checkpoint time and automatically imported from the IMSRSC repository into an IMS system at IMS cold start.

Related concepts

“Maintaining MSC resources in the IMSRSC repository” on page 734
Maintain MSC resource definitions in the IMSRSC repository to store the definitions in a single, centralized location for all members of an IMSplex. Maintaining MSC resource definitions in the IMSRSC repository also enables the definitions to be saved across an IMS cold start.

Related tasks

Defining the IMSRSC repository (System Definition)

Related reference

MSC section of the DFSDFxxx member (System Definition)

DYNAMIC_RESOURCES section of the DFSDFxxx member (System Definition)

COMMON_SERVICE_LAYER section of the DFSDFxxx member (System Definition)

REPOSITORY section of the DFSDFxxx member (System Definition)

Enabling MSC with the MSC= execution parameter

You enable the Multiple Systems Coupling (MSC) feature in an IMS system by specifying MSC=Y in the DFSPB*xxx* member of the IMS PROCLIB data set.

If you are enabling dynamic definition for MSC resources, the IMS system must configured to support dynamic definitions, with features such as the CSL Operations Manager (OM) and the Structured Call Interface (SCI) enabled. You also need a type-2 command interface such as the TSO SPOC. If you are using the IMSRSC repository, the Resource Manager (RM), the Repository Server, and the Common Queue Server must also be enabled.

When MSC is enabled in an IMS system by coding the MSC=Y execution parameter, MSC resources do not need to be defined by using stage-1 system definition macros before IMS is started. If MSC resources are defined during IMS system definition, the MSC resources are loaded during IMS startup.

Recommendation: To ease operations when dynamic MSC is enabled and MSC resources are no longer defined using the system definition process, specify the MSC=Y execution parameter, even if MSC resources are defined with the system definition process.

You might enable MSC without defining any MSC resources during system definition in the following cases:

- The MSC resources are already defined in the IMSRSC repository
- You plan to create MSC resources by using IMS type-2 CREATE commands. In this case, the dynamic definition of MSC resources must be enabled.
- The IMS system is a back-end IMS system in a shared-queues group that does not require any MSC links. MSC is being enabled only to process messages on the shared queue that were previously transmitted on an MSC link.

To enable MSC:

1. Define one or more local system IDs (SYSIDs) for the IMS system by using the SYSID parameter in the MSC section of the DFSDF*xxx* member of the IMS PROCLIB data set.

   If one or more MSNAME macros are included in the IMS system definition, coding the SYSID parameter is optional.
2. Optionally, enable dynamic definition of MSC resources by specifying MSCRSCS=DYN in the MSC section of the DFSDF*xxx* member.

   Dynamic definition of MSC resources must be enabled to create or delete MSC resources by using IMS type-2 commands.
3. If MSCRSCS=DYN, specify your options for importing your MSC resource definitions on the AUTOIMPORT parameter in the DYNAMIC_RESOURCES section of the DFSDFxxx member.
   - To have IMS automatically detect and load your MSC resource definitions during startup, regardless of how they were defined and where they are stored, specify AUTOIMPORT=AUTO. AUTOIMPORT=AUTO is the default.
   - To have IMS load MSC resources from only the IMSRSC repository, specify AUTOIMPORT=REPO. When AUTOIMPORT=REPO, IMS imports both MSC and MODBLKS resources from the IMSRSC repository.
   - To have IMS load only MSC resources that are defined during IMS system definition, specify AUTOIMPORT=MSCGEN.
   - To prevent loading any MSC resources during startup, specify AUTOIMPORT=NO. The MSC resource definitions will need to be either imported later from the IMSRSC repository by using the IMPORT DEFN command or defined by using IMS type-2 CREATE commands.
4. Enable MSC by specifying the MSC=Y execution parameter in your startup procedure.
5. Cold start IMS.

   If any MSC resources were generated into the DFSCLL3x member of the IMS.SDFSRESL data set during system definition, IMS loads them into the online system during startup.

Related reference

MSC= parameter for procedures (System Definition)

DFSDFxxx member of the IMS PROCLIB data set (System Definition)

Enabling MSC during system definition

You can enable MSC during system definition by defining at least one MSC link with the MSPLINK, MSLINK, and MSNAME stage-1 system definition macros.

When MSC resources are defined by macros during stage-1 system definition, you do not need to specify the MSC= execution parameter.

Recommendation: If you are enabling MSC in a back-end IMS system in a shared-queues environment solely for the purpose of processing messages that were received on the front-end from an MSC link, do not use stage-1 system definition macros to enable MSC in the back-end IMS systems. Instead, specify the MSC=Y execution parameter in the startup procedure for the IMS system. The back-end IMS systems are easier to clone when MSC is enabled by the MSC execution parameter instead of stage-1 system definition macros.

Related reference

MSC= parameter for procedures (System Definition)

MSPLINK macro (System Definition)

MSLINK macro (System Definition)

MSNAME macro (System Definition)

Defining a SYSID

System identifiers (SYSIDs) are 2-byte numbers between 1 - 2036 that identify the IMS systems in an MSC network.

You can assign a local SYSID to an IMS system in the following ways:

- MSNAME stage-1 system definition macro
- MSC section of the DFSDFxxx member of the IMS PROCLIB data set
- IMS type-2 command **CREATE MSNAME**

You can change SYSIDs online by using the IMS type-2 command UPDATE MSNAME NAME(*msname*) SET(SIDL | SIDR).

Related concepts

"System identifiers (SYSIDs)" on page 677
MSC uses system identifiers (SYSIDs), two-byte numbers between 1 - 2036, to identify the IMS systems in an MSC network.

Related reference

MSNAME macro (System Definition)

MSC section of the DFSDFxxx member (System Definition)

CREATE MSNAME command (Commands)

Disabling MSC with the MSC= execution parameter

You can disable the Multiple Systems Coupling (MSC) feature in an IMS system by specifying MSC=N in the IMS execution parameters and cold starting IMS.

When MSC=N is specified, no MSC resources or control blocks are loaded during IMS initialization. Any MSC parameters that are specified in the DFSDFxxx or DFSDCxxx PROCLIB members are ignored. Any MSC definitions in the stage-1 system definition input are also ignored.

Alternatively, you can also disable MSC by removing all MSC definitions from the IMS system definition stage-1 input and omitting the MSC= execution parameter. When no MSC stage-1 system definition input is present, MSC=N is the default.

To disable MSC:

1. Specify MSC=N in the IMS execution parameters.

   The MSC= parameter can be specified in the IMS or DCC startup procedure in the DFSPBxxx member of the IMS PROCLIB data set.
2. Cold start IMS.
3. Confirm that MSC is disabled by checking the MSC= specification in the final DFS1929I message that is issued at the end of startup.

Local system definitions

When you define an MSC network, you need to define transactions, logical terminals, and physical and logical connections in each local system. You can define these resources by using system definition macros, or you can define them dynamically by using IMS type-2 CREATE commands for the resource type you are defining.

More specifically, for each local system you need to define:

- All transactions entered or processed by that system
- All logical terminals that are attached to that system and all logical terminals in remote systems that are referenced by transactions processed in that system, or by terminal operators, unless either a Program Routing exit routine is used or MSC directed routing is used

- The physical and logical connections between that system and the remote systems that share in the processing of the specified transactions

The system definition macros that you need to prepare in each system are summarized in the following table. The first macro that you use is IMSCTRL. Using the MSVID keyword, you assign a number in the range from 1 to 255 as a unique identifier of that system. This causes a control block, DFSMSxxx, to be built for use with the Multiple Systems Verification utility. (The 3-digit suffix, *xxx*, matches the MSVID parameter; one unique control block exists for each system in the MSC network.) This offline utility helps verify that system definitions for all partner systems are consistent. The IMSMSV procedure is generated in IMS.SDFSPROC in order to execute this utility.

Resource	Identification	Macro	Number coded
System	System ID	IMSCTRL	1
Programs	PSB name	APPLCTN	1 per PSB
Local transactions	Transaction code	TRANSACT	1 or more per APPLCTN
Remote transactions	Transaction code	TRANSACT	1 or more per APPLCTN in a remote system
Physical connections	Physical link name	MSPLINK	1 per connection
Logical connections	Logical link name	MSLINK	1 or more per MSPLINK, depending on the physical link type"1" on page 694
Routing names	Logical path name	MSNAME	1 or more per MSLINK
Local terminals	LTERM name	NAME	1 per local terminal
Remote terminals	LTERM name	NAME"2" on page 694	1 per remote terminal

Notes:

1. Multiple-session TCP/IP physical links and VTAM physical links have additional MSLINK macros. Channel-to-channel (CTC) and memory-to-memory (MTM) physical link types can have only one logical link per physical link.
2. Define with an ETO MSC descriptor if the corresponding local terminal is an ETO terminal.

Related reference

Macros used in IMS environments (System Definition)

Multiple Systems Verification utility (DFSUMSV0) (System Utilities)

Defining partner systems

In order to define a usable link between two partner systems, an MSPLINK, MSLINK, and MSNAME must be defined on both sides. These MSC resources can be defined dynamically by using CREATE MSPLINK, CREATE MSLINK, and CREATE MSNAME commands, or statically by using stage-1 system definition macros MSPLINK, MSLINK, and MSNAME.

A relationship exists between the three MSC resource types MSPLINK, MSLINK, and MSNAME that define the names of connections between systems and the logical names that are used in commands.

Defining the physical link

You can define different types of physical links to connect IMS systems. Your choice depends on the hardware that is required for each of the links.

The physical link choices are:

Channel-to-channel (CTC)

Memory-to-memory (MTM)
TCP/IP
VTAM

You can use either the CREATE MSPLINK type-2 command or the MSPLINK stage-1 system definition macro to define the physical link and its attributes.

To use the CREATE MSPLINK command, dynamic resource definition must be enabled in your IMS system. The CREATE MSPLINK command is logged.

If dynamic definition is not enabled in your IMS system, you must use the MSPLINK macro to define your physical links. During system definition, define all physical connections that might potentially be used, even if several might be for backup purposes or might not be intended for continual use.

You must declare the physical link for partner systems in both system definitions. Using the TYPE keyword, declare the kind of physical link to use.

Assign a name to the physical link. The name of the physical link is used for logical link definition to match a connection between systems with the physical device or transmission technique that is to be used. The name is also used to identify the link when issuing commands.

If part of a conversation is to be passed from one system to another, use the maximum scratchpad area (SPA) size, if that SPA size exceeds the message segment size.

You can change the attributes of existing physical links by using the type-2 **UPDATE MSPLINK** command. Unless you save the physical link to the IMSRSC repository, the changes remain in effect only until the next cold start of the IMS system.

Related reference

MSPLINK macro (System Definition)
Macros used in IMS environments (System Definition)
CREATE MSPLINK command (Commands)
QUERY MSPLINK command (Commands)
UPDATE MSLINK command (Commands)
UPDATE MSPLINK command (Commands)

Buffer sizes for physical links

The physical link buffer size defines the default buffer size for all logical links assigned to the physical link.

Use at least the size of the largest message segment that is to be transmitted across this physical link. The buffer sizes at each end of a physical link must be equal.

Specify a buffer size with the BUFSIZE keyword on either the CREATE MSPLINK command or the MSPLINK macro.

You can specify different buffer sizes for individual logical links by using either the IMS type-2 command **UPDATE MSLINK** NAME(*linkname*) SET(BUFSIZE(*new_bufsize*)) or the IMS type-1 command **/UPDATE MSLINK** NAME(*linkname*) SET(BUFSIZE(*new_bufsize*)).

To update the buffer sizes for all logical links that are assigned to a physical link, use the command **UPDATE MSPLINK** NAME(*linkname*) SET(BUFSIZE(*new_bufsize*)).

See the description of the BUFSIZE keyword for the minimum and maximum sizes that you can specify for link buffers.

You can display MSC statistics to help determine the optimum buffer size. MSC statistics are displayed by the IMS type-2 command **QUERY MSLINK** NAME(*linkname*) SHOW(STATISTICS). Use the statistics Hi_Msg_Send_SZ and Hi_Msg_Rec_SZ to see the maximum message sizes being sent and received. You can also compare the number of send and receive I/O requests to the number of messages that are sent and received. If IMS requires multiple send and receive I/O requests to send a single message, the buffers are too small. The send and receive I/O requests are captured by the statistics Tot_Send_CT and Tot_Rec_CT. The number of messages sent and received are captured by the statistics Tot_Msg_Send_CT and Tot_Msg_Rec_CT.

Depending on the physical link type, storage for the link buffer sizes are allocated from different pools. TCP/IP and VTAM link buffers are allocated from the high input/output pool (HIOP). CTC link buffers are allocated from the communications input/output pool (CIOP). MTM link buffers are allocated from subpool 231 in the common storage area (CSA).

Related reference

MSPLINK macro (System Definition)
Macros used in IMS environments (System Definition)
CREATE MSPLINK command (Commands)
QUERY MSLINK command (Commands)
UPDATE MSLINK command (Commands)
UPDATE MSPLINK command (Commands)

Defining a CTC physical link

The CTC adapter is a channel-to-channel hardware connection.

CTC links are usually used only when the IMS systems are in the same data center. You can assign only one logical link to a physical link that uses the CTC connection type.

You can use either the CREATE MSPLINK type-2 command or the MSPLINK stage-1 system definition macro to define the physical link and its attributes.

To use the CREATE MSPLINK command, dynamic resource definition must be enabled in your IMS system. The CREATE MSPLINK command is logged.

If dynamic definition is not enabled in your IMS system, you must use the MSPLINK macro to define your physical links. During system definition, define all physical connections that might potentially be used, even if several might be for backup purposes or might not be intended for continual use.

For CTC links, you must add an address parameter value. Each CTC link must have a unique address.

If you create a CTC link by using the MSPLINK system definition macro, you must also add a DD name.

Related reference

MSPLINK macro (System Definition)
Macros used in IMS environments (System Definition)
CREATE MSPLINK command (Commands)
QUERY MSPLINK command (Commands)
UPDATE MSLINK command (Commands)
UPDATE MSPLINK command (Commands)

Defining an MTM physical link

The MTM link is a software link between IMS subsystems that are running in the same logical partition.

You can assign only one logical link to a physical link that uses the MTM connection type.

You can use either the CREATE MSPLINK type-2 command or the MSPLINK stage-1 system definition macro to define the physical link and its attributes.

To use the CREATE MSPLINK command, dynamic resource definition must be enabled in your IMS system. The CREATE MSPLINK command is logged.

If dynamic definition is not enabled in your IMS system, you must use the MSPLINK macro to define your physical links. During system definition, define all physical connections that might potentially be used, even if several might be for backup purposes or might not be intended for continual use.

Related reference

MSPLINK macro (System Definition)
Macros used in IMS environments (System Definition)
CREATE MSPLINK command (Commands)

QUERY MSPLINK command (Commands)
UPDATE MSLINK command (Commands)

Defining a TCP/IP physical link

For TCP/IP physical links, the connection between the two IMS systems uses the TCP/IP connection and networking protocols that are used by the internet.

Usually used when the IMS systems are in different data centers. IMS Connect manages the TCP/IP connections and protocols for the physical links.

Communications between MSC and IMS Connect are managed by the Structured Call Interface (SCI) component of the IMS Common Service Layer (CSL) in an IMSplex.

You can assign multiple logical links to a physical link that uses the TCP/IP connection type. TCP/IP physical links always operate in MSC bandwidth mode and require a slightly larger buffer size than the other physical link types.

You can use either the CREATE MSPLINK type-2 command or the MSPLINK stage-1 system definition macro to define the physical link and its attributes.

To use the CREATE MSPLINK command, dynamic resource definition must be enabled in your IMS system. The CREATE MSPLINK command is logged.

If dynamic definition is not enabled in your IMS system, you must use the MSPLINK macro to define your physical links. During system definition, define all physical connections that might potentially be used, even if several might be for backup purposes or might not be intended for continual use.

For physical links that use TCP/IP, the specification on the NAME keyword of the MSPLINK macro must match the specification on the IMSID keyword of the IMSCTRL system definition macro of the remote IMS system.

The SESSION parameter indicates the number of parallel sessions, or logical links, that can be active for TCP/IP and VTAM physical link types.

Related reference
MSPLINK macro (System Definition)
Macros used in IMS environments (System Definition)
CREATE MSPLINK command (Commands)
QUERY MSPLINK command (Commands)
UPDATE MSLINK command (Commands)
UPDATE MSPLINK command (Commands)

Defining a VTAM physical link

A VTAM physical link is supported by SNA VTAM.

You can use either the CREATE MSPLINK type-2 command or the MSPLINK stage-1 system definition macro to define the physical link and its attributes.

To use the CREATE MSPLINK command, dynamic resource definition must be enabled in your IMS system. The CREATE MSPLINK command is logged.

If dynamic definition is not enabled in your IMS system, you must use the MSPLINK macro to define your physical links. During system definition, define all physical connections that might potentially be used, even if several might be for backup purposes or might not be intended for continual use.

For physical links that use VTAM, you must supply a NAME keyword that matches the label on the VTAM APPL statement for the remote system. The NAME keyword and the VTAM APPL statement both specify the VTAM node name.

The label on the VTAM APPL statement also serves as a default value for the ACBNAME parameter on the same statement. Regardless of whether you use the default value for the ACBNAME parameter, this value must match the APPLID parameter on the IMS COMM macro.

The SESSION parameter indicates the number of parallel sessions, or logical links, that can be active for TCP/IP and VTAM physical link types.

Related reference

MSPLINK macro (System Definition)
Macros used in IMS environments (System Definition)
CREATE MSPLINK command (Commands)
QUERY MSPLINK command (Commands)
UPDATE MSLINK command (Commands)
UPDATE MSPLINK command (Commands)

Defining the logical link

You can define the logical link definition statically by using the MSLINK system definition macro or dynamically by using the type-2 CREATE MSLINK command. Both methods enable you to name the link, associate the logical link with a logical link defined in a partner system, and define the type of physical link the logical link can be used with.

You can assign a name to the logical link by using the macro label field. Logical link names are used by the type-2 commands **QUERY MSLINK** or **UPDATE MSLINK** to identify the target logical link. If you do not define a logical link name using a macro, IMS assigns the default logical link name DFSL*xxxx*, where *xxxx* is the logical link number.

You associate the logical link with the logical link defined in a partner system by specifying a 2-character alphanumeric partner ID on the PARTNER keyword. The partner ID that you specify must match a corresponding partner ID specified in an MSLINK macro in the partner system. The matching values of the PARTNER keyword represent a logical connection between the two partner systems. For example, if B1 is defined as the partner ID in one logical link, this same B1 partner ID must be specified in the corresponding MSLINK macro of the partner system.

You match the type of physical connection that can potentially be used with this logical link by specifying the MSPLINK keyword. Depending on the type of physical link you specify, you can assign one or more MSLINK macros to the physical link:

- When the physical link type is CTC or MTM, one MSLINK macro exists for each kind of physical connection.
- When the physical link type is TCP/IP or VTAM, multiple logical links can be assigned to one physical link. For example, System_B and System_C can have two logical links, one using MTM and the other CTC; System_C and System_D can have two logical links, both of which use one TCP/IP or VTAM physical link.

You can define a maximum of 999 logical links for each IMS system.

You can also define the logical link by using the type-2 CREATE MSLINK command instead of using a macro.

You can change logical link attributes that are defined by the MSLINK macro by using the **UPDATE MSLINK** command. The attributes you can change include the name, the physical link assignment, the partner ID, the buffer size, and others. The changes remain in effect until the next cold start of the IMS system.

Related reference

CREATE MSLINK command (Commands)
UPDATE MSLINK command (Commands)
MSLINK macro (System Definition)

Defining a logical path

When the operator connects two systems, a choice of physical connection type might be possible. Therefore, the logical link name is used. You can define that logical path name by using the MSNAME

system definition macro or you can define them dynamically by using the type-2 CREATE MSNAME command.

The SYSID keyword is used to declare the two systems that are joined in a pathway. You select the 1-digit identifiers from the range 1 to 2036, one for the remote system and the other for the local system. For example, (1,3) specifies that any message using this path is being sent to the remote system number 1, and that the local system number is 3.

You can change the remote and local system IDs of a logical path by using the type-2 command **UPDATE MSNAME**. The changes will remain in effect only until the next cold start of the IMS system.

The MSNAME macro can be followed by a set of NAME macros on which you can specify the LTERM names for terminals that are in remote systems. You do not need to declare every terminal in the remote system that is entering transactions, but only those that enter traffic destined for this local system. If the LTERM in the remote system is for an ETO terminal that enters transactions destined for this system, define the LTERMs using ETO MSC descriptors instead of NAME macros.

Related reference

CREATE MSNAME command (Commands)

MSNAME macro (System Definition)

Setting link priorities for remote transactions

You can assign priorities to remote transactions by using the PRTY= keyword on the TRANSACT macro during system definition in the destination system. You can also assign priorities dynamically by using the type-2 commands CREATE TRAN and UPDATE TRAN.

You might want to assign priorities because a remote input transaction might have a different scheduling priority in the destination system from that defined for it during system definition of the originating system. This means there could be a long wait for a response, even though the remote transaction had a high priority when it was defined with the SYSID= parameter of the TRANSACT macro or the NPRI keyword of the CREATE TRAN command.

For example, in the following figure, Terminal 1 is sending a remote transaction (ASMB1) to System B.

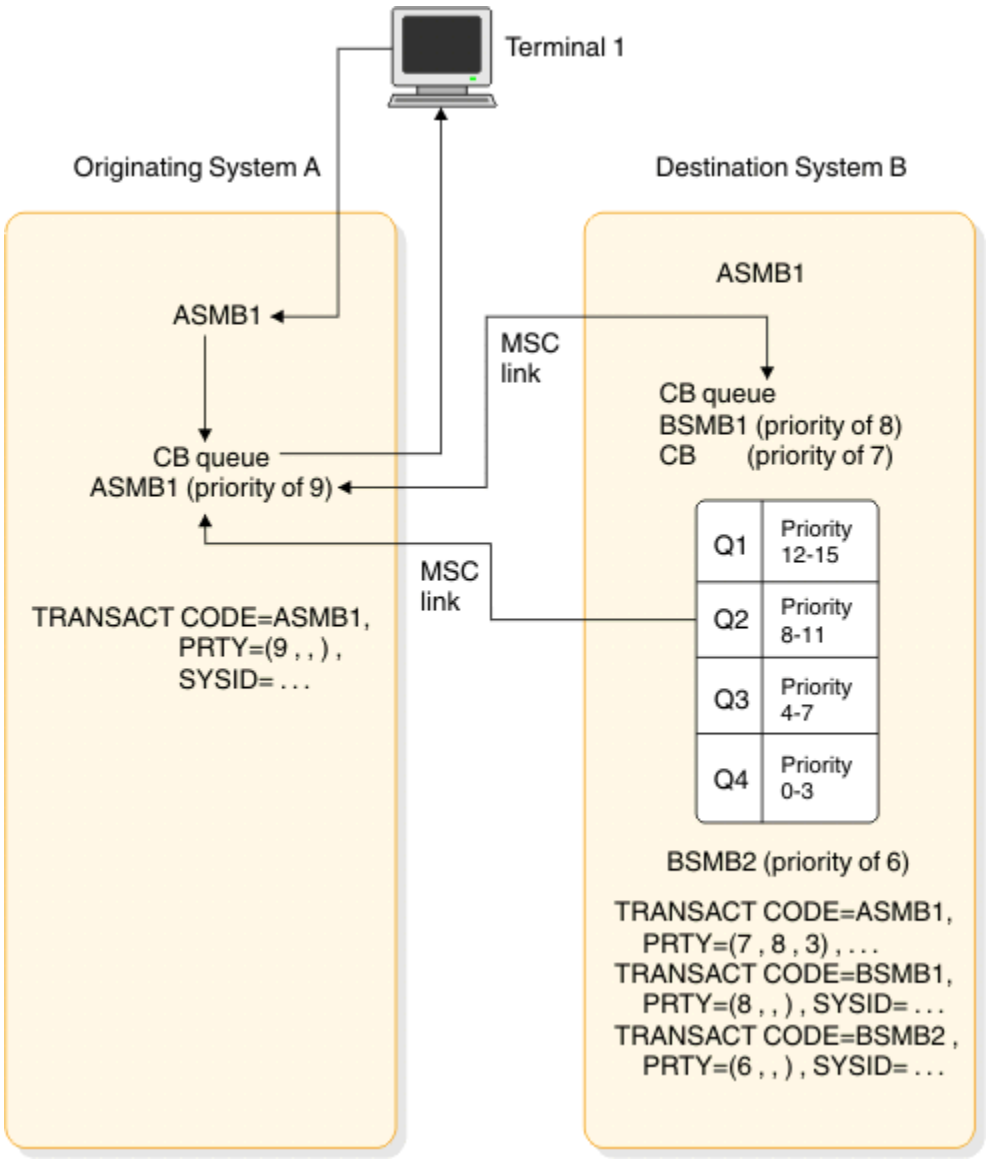


Figure 121. Link priorities for a remote transaction

In IMS system A, this transaction is enqueued to a control block (CB) and has a priority of 9 (defined with the SYSID= parameter). When ASMB1 is sent to system B, it becomes a local transaction and is processed. When the message router is called to send a response, it enqueues the response on the control block that represents the MSC link. If ASMB1 is defined in system B with a priority of 7, all transactions with priorities of 12-15 (in queue 1) are processed first, and 8-11 (in queue 2) are processed next. The ASMB1 (with a priority of 7 in queue 3) is processed next. However, as shown in the figure, the limit priority can be set at 8 and the limit count can be set at 3. As a result, when the number of message queues to be processed reaches 3, the priority of ASMB1 is changed to 8, and subsequent responses are placed on queue 2. On system B, all messages for BSMB1 are sent before responses are sent. Messages for BSMB2 are sent last in this example.

In a shared-queues environment, however, priority applies only to messages that are received from the MSC link to be processed. Priority does not apply to response messages (messages going back to the input system) because the coupling facility does not have the four queues. Instead, all response messages are sent FIFO.

Related reference

CREATE TRAN command (Commands)
UPDATE TRAN command (Commands)

TRANSACT macro (System Definition)

Serial transaction processing in an MSC network

Serial transactions are processed in the order they are received in relative to other transactions of the same type.

You can ensure serial processing of transactions in remote MSC IMS systems by taking the following steps:

- Define the transaction as serial in both the local and remote MSC IMS systems.
- Restrict the transactions to a single logical link path between the local and remote MSC IMS systems.
- Send all serial transactions of the same type to the same remote MSC IMS for processing.

Serialization is not preserved for transactions that:

- Are sent across different logical link paths
- Originate from different MSC IMS systems
- Are processed in different remote MSC IMS systems

The PRTY= keyword and output messages of serial transactions in an MSC network

The serial processing of the output messages of a serial transaction can become unpredictable in an MSC network if the *normal* and *limit* parameters of the priority (defined by the CREATE TRAN SET(NPRI(*normalpriority*),LPRI(*limitpriority*)) or the TRANSACT macro PRTY= keyword) are not equal.

If the *normal* and *limit* parameters are not equal and the number of output messages on a message queue becomes equal to or greater than the value of the *limit_count* parameter of the PRTY= keyword, output messages received by a remote MSC IMS system after the *limit_count* value has been reached might be processed before messages received by the remote MSC IMS system.

Related reference

CREATE TRAN command (Commands)

UPDATE TRAN command (Commands)

TRANSACT macro (System Definition)

Specifying exit routines

These topics describe how to specify exit routines that are used with MSC.

Related concepts

“TM and MSC Message Routing and Control user exit routine overview” on page 730

Message routing is automatic, according to the defined scheme, unless you use the TM and MSC Message Routing and Control user exit routine (DFSMSCE0), which provides routing options and control of messages.

Routing messages with DFSMSCE0

The TM and MSC Message Routing and Control User exit routine (DFSMSCE0) is not included in the system definition process. Therefore, incorporate DFSMSCE0 in IMS.SDFSRESL.

The terminal routing entry points are invoked in the input system, possibly changing the destination. When transactions arrive at an IMS across the MSC link, the link-receive entry points can be invoked. These entry points can change the destination again. Because destinations can be changed at so many different points, clearly document how the destinations are changed, and explain the reasons for the new destinations, which are used for processing. You might need to correlate the transaction code changes as information for master terminal operators so that they can interpret a display of queue status.

If the DFSMSCE0 exit routine is used in an IMSplex environment with shared queues, the DFSMSCE0 exit routine can be used in the IMS systems within the IMSplex for affinity routing, which establishes an affinity between a transaction message and a back-end IMS system. Affinity routing effectively

suppresses the sharing of the transaction message on the shared queues so that only IMS systems that have affinity are notified when the transaction arrives on the shared queues.

DFSMSCE0 IMSplex affinity routing is also a way to route APPC synchronous transaction or an OTMA send-then-commit (CM1) transaction to back-end IMS systems in an IMSplex for processing when either RRS=N or AOS=N. Without assigning affinity, these APPC and OTMA transactions cannot be routed to back-end systems for processing. MSC must be enabled in the IMSplex to use DFSMSCE0 IMSplex affinity routing with APPC synchronous or OTMA send-then-commit (CM1) transactions.

Be aware of how transactions can be routed for different stages in the total processing. If input from IMS A is being processed in IMS C, the processing program can invoke the exit routine to determine whether to send its output to the original input terminal or to another location. Your documentation of the exit routine should show the patterns of alternative message destination. An end user could be expecting output and be unaware that it is being sent to another component.

Related concepts

"TM and MSC Message Routing and Control user exit routine overview" on page 730
Message routing is automatic, according to the defined scheme, unless you use the TM and MSC Message Routing and Control user exit routine (DFSMSCE0), which provides routing options and control of messages.

Related reference

TM and MSC Message Routing and Control User exit routine (DFSMSCE0) (Exit Routines)

Managing error messages with DFSCMUX0

The Message Control/Error exit routine (DFSCMUX0) allows you to manage and control messages that are in error on an MSC link or on a message queue. It is activated by an MSC link start, link termination, send error, or receive errors.

The Message Control/Error exit routine is called for these situations for APPC/IMS:

- If an LU 6.2 session fails while sending an output message to an LU 6.2 program
- If a send to an LU 6.2 application program is rejected with a `Deallocate with Send_Error` message
- When **/DEQUEUE** *luname tpname* is entered

You can customize the Message Control/Error exit routine to ask IMS to take any of these actions:

- Take the default action: discard the message and proceed with the **/DEQUEUE** command.
- Discard the message in error.
- Discard the message in error and notify the MTO or originating terminal of this error.
- Re-route the message in error to a different local or remote transaction, a local or remote LTERM, or an LU 6.2 application program.

The sample exit routine that is provided in IMS.ADFSSRC uses the default action.

Related reference

Message Control/Error exit routine (DFSCMUX0) (Exit Routines)

How network definition is affected by multiple systems

Each system definition must have all terminals that are connected to it defined during system definition using either the TERMINAL and NAME macros or defined dynamically using the type-2 commands CREATE LTERM and CREATE MSNAME. However, if terminals in other systems are capable of sending messages that are destined for the defined system, those terminals are logically part of the network for that system.

You can define your MSC physical and logical links either dynamically with CREATE MSPLINK and CREATE MSLINK commands, or statically through the system definition process. You can define your MSC remote

LTERMs either dynamically with the CREATE LTERM command, or statically with the stage-1 system definition NAME macro, or during IMS initialization with Extended Terminal Facility (ETO).

When a transaction is processed in a remote system, the input LTERM name in the local system is carried over as part of the message. If the processing program uses the alternate PCB to direct a message to another terminal besides the input terminal, those destinations need to be declared as remote, unless directed routing is used. Define the LTERM names for all inputs with NAME macros or with CREATE LTERM commands. Position the NAME macros in a group after the MSNAME macro. You now have a set of LTERMs that collectively can occur in several system definition decks. For example, TERMA can be present in the input system, in the intermediate system, and in the processing system.

When planning for the network, keep in mind that message queues for an input system or an intermediate system must allow for the remote transactions being queued. When allocating space for the message queues take into account the message lengths and their expected loads. In a similar way, you must allow for the presence of these messages in I/O buffers, even when they are not going to be processed in that system.

Related tasks

"Administering the Extended Terminal Option" on page 69
The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

Related reference

CREATE LTERM command (Commands)
CREATE MSNAME command (Commands)
NAME macro (System Definition)
TERMINAL macro (System Definition)

Verifying transaction definitions across systems

After you have completed system definitions for several systems that are to be part of a multiple-system network, the transaction codes, LTERM names, and system identification numbers are all identifiable in control blocks.

Using the multiple systems verification utility

You can use the IMS Multiple Systems Verification utility offline in order to verify that the names you have used are consistent across system definitions.

Example: If you use TRANX as a code in two definitions but TRANXX in a third system, the Multiple Systems Verification utility sends you a message highlighting the single reference to TRANXX.

You can execute the Multiple Systems Verification utility by using the IMSMSV procedure in IMS.PROCLIB.

The Multiple Systems Verification utility also checks to ensure that transaction codes are not defined as local in more than one system.

The Multiple Systems Verification utility uses input control statements that specify the system identification numbers that you want to include in the checking. Its output is in the form of a multisystem path map.

Restriction: The Multiple Systems Verification utility cannot detect errors caused by improper use of MSC directed routing.

Related reference

Multiple Systems Verification utility (DFSUMSV0) (System Utilities)

Verifying the system definition status online

You can check the consistency of your system definitions or ETO MSC descriptors using the **/MSVERIFY** command when each of the component systems of your network is operational.

Recommendation: Use the **/MSVERIFY** command during the system test phase, rather than during production. The **/MSVERIFY** command generates considerable traffic.

The **/MSVERIFY** command validates that:

- Logical terminals exist for the remote LTERMs defined.
- Transactions are defined and have the same attributes in their remote system as in the local system.
- Logical path definitions are consistent and usable.

Recommendation: Define your remote LTERMs using ETO MSC descriptors. Otherwise, the Multiple Systems Verification utility recognizes remote LTERMs, but not the corresponding local LTERM in the target system.

The **/MSVERIFY** command verifies consistent definition between the local system and one remote system for each command. Verification begins with local SYSIDs and then proceeds to remote SYSIDs. As each segment of the processing begins, the MTO is notified of the local and remote SYSIDs to be verified. This message, DFS2234I or DFS2236I, also is time-stamped. As the specific local and remote SYSIDs are verified, the MTO receives verification-completed notifications with corresponding time stamps. If a definition or assignment error is discovered, error messages are returned.

During command processing, the local system sends all its own locally defined MSC elements (SYSIDs) to the remote IMS system specified in the **/MSVERIFY** command. The remote system responds by returning all destinations and their attributes that belong to the local system (as the remote system definition or assignments have them described). The command input system checks local and remote definitions for consistency.

The response list should be checked for completeness using the listing of SYSIDs that are to be verified given in the DFS2234I or DFS2236I `IN PROGRESS` messages and the correspondence of time stamps in responses.

The local system then sends all MSC elements to the remote system expecting them to be defined in that remote system with corresponding attributes (as the local system has them defined). The remote system performs consistency checks and routes error messages to the command input terminal.

MSC path consistency is only checked for current operational assignments of logical link paths to logical links, and logical links to physical links.

Recommendation: Ensure that the MSC definition and assignments are accurate, even when receiving favorable responses (that return no errors). Definition and assignment errors can prevent the return of a command response for some SYSIDs. Regard the absence of a response for a particular SYSID as significant.

Reviewing these error responses from the **/MSVERIFY** command might help you to avoid these definition and assignment errors:

```
DFS2235I SYSID __ is defined as local in both systems
DFS2241I __ is defined as remote transaction in both systems
DFS2242I __ is not defined as LTERM in both systems
DFS2243I __ is not defined as transaction in both systems
DFS2245I Multisegment transaction flag for __ not consistent
DFS2246I Non-inquiry only flag for __ not consistent
DFS2247I Conversational flag for __ not consistent
DFS2248I Irrecoverable flag for __ not consistent
DFS2249I Fixed length SPA flag for __ is not consistent
DFS2250I The SPA length for __ is not the same
```

The **/MSVERIFY** command cannot detect errors caused by improper use of MSC directed routing.

Security considerations for MSC

For those transactions that are processed in another system, perform as much security checking as is required for the primary message. RACF can be used to protect IMS resources in an MSC network.

Signon verification, combined with transaction authorization and password checking, allows you to control the processing at input time. The resource definition to RACF must declare the transaction name, even when the transaction is not processed in the system where the security tables are built.

Security controls in an MSC network are performed independently in each local and remote IMS. An intermediate IMS in an MSC environment, which is neither local nor remote for a given transaction, does not perform any security checking for that transaction.

RACF can provide transaction authorization checking when a destination system receives a message to process. The amount of checking that RACF provides depends on the MSCSEC= parameter in the DFSDCxxx IMS.PROCLIB member and on feedback from the DFSMSCE0 exit routine. The DFSMSCE0 user exit can optionally override or accept the system DFSDCxxx member security, on a message by message basis.

Transactions received in a remote IMS on an MSC link are passed to the transaction authorization module for authorization checking, but because the password is not passed across the link, transaction authorization checking fails if a password is required. Transactions that do not require a password can be accepted.

To allow a transaction to be scheduled in a remote destination IMS, you can authorize its processing with resource access security (RAS). To use RAS security, the transaction must be defined to RACF as authorized for use by the dependent region.

If the RACF security environment is not available in the destination system (as when a `/SIGN ON` command is entered with RACF), the security environment will be dynamically created to allow the transaction authorization to proceed.

Related concepts

IMS security (System Administration)

Operations for Multiple Systems Coupling

Each system in an MSC network is operationally an independent unit. Each system exclusively owns its own communication resources, which are controlled by its own master terminal.

In many cases, the MSC operations performed in one IMS system require corresponding operations to be performed in a partner IMS system.

Common operational tasks include starting and stopping communications on MSC links, modifying link assignments, and recovery.

Related concepts

MSC operations (Operations and Automation)

MSC link statistics

IMS maintains statistics for each MSC logical link. You can use link statistics for tuning MSC systems to process MSC messages more efficiently.

IMS records three types of MSC logical link statistics:

General statistics
Statistics start time, MSC ITASK dispatch counts, MSC ITASK processing times, and the rate and number of logger check writes

Send statistics
Messages sent, byte count sent, send message sizes, queue manager get counts and times, and send I/O times

Receive statistics
Messages received, byte count received, receive message sizes, QMGR insert counts and times, and receive I/O times

The I/O times for sends and receives are kept only when MSC bandwidth mode is enabled. If MSC bandwidth mode is disabled, the values for the I/O statistics are zero.

You can view the current statistics for a logical link by issuing the type-2 command `QUERY MSLINK NAME(linkname) SHOW(STATISTICS)`.

IMS maintains two copies of the statistics for each logical link: the running cumulative totals since the last start or restart of IMS and the running cumulative totals since the last checkpoint or manual reset of the link statistics. The command `QUERY MSLINK NAME(linkname) SHOW(STATISTICS)` shows only the statistics since the last checkpoint or manual reset, unless the link statistics have not been reset since the last restart of IMS. Otherwise, you can see the running totals since the last IMS restart only by looking at the X'4513 ' log records.

You can reset the statistics for a link manually or have IMS automatically reset the statistics for a link at each system checkpoint. Resetting link statistics can be useful when running performance tests and comparing results. To manually reset the statistics for a link or to turn on or turn off automatic checkpoint reset for link statistics, issue the type-2 command `UPDATE MSLINK NAME(linkname) START(STATISTICS) OPTION(reset_option)` with the appropriate reset option. The logical link does not need to be stopped or idle to reset link statistics or to change from automatic to manual reset mode and back again.

Resetting MSC logical link statistics does not affect the logging of MSC logical link statistics. MSC logical link statistics are always logged and cannot be reset.

Related tasks
Diagnosing link problems by using MSC link statistics (Diagnosis)

Benchmark link activity

You can benchmark link processing by documenting what the normal and high levels of message activity are for your MSC logical links by querying the link statistics.

If you document link activity levels, you can then use them to help recognize problems when current activity levels differ from the documented levels, or you can compare previous levels to current levels to see if more logical links or network capacity is needed to ease capacity limitations or bottlenecks.

Determine your optimum MSC link type

You can determine the specific resource requirements of a message by resetting the logical link statistics, sending one typical message, and recording the statistics.

A single message represents an MSC unit of work, and the resulting statistics for the single message represent the resources an MSC unit of work require, such as time, calls, and so forth.

You can also determine the average resource requirements for MSC units of work by sending multiple messages and then dividing the resulting cumulative statistics by the number of messages sent.

After you know how much resources a unit of work requires on a specific link type, if your installation uses multiple physical link types (that is, CTC, MTM, TCP/IP, and VTAM), you can compare the resource requirements of units of work on each type of link, and then use the link type that utilizes the least amount of resources.

Reset statistics regularly at system checkpoint

Unless you have a specific reason for resetting MSC logical link statistics manually, you should configure IMS to reset link statistics regularly at system checkpoint by specifying a reset option of `RESET,CHKPT` on the type-2 command `UPDATE MSLINK`.

Automatic reset at system checkpoint is the default reset option that is in effect when IMS restarts.

The longer the interval between resets, the less useful the statistics become for determining problems and other analyses. If you need to run a benchmark or gather statistics for an interval longer than the interval between IMS system checkpoints, you can turn off the automatic reset at checkpoints by issuing the UPDATE MSLINK command with the NORESET,CHKPT parameters.

When automatic resets at checkpoint are turned off, IMS does not reset link statistics, and you must manually reset the statistics to start a new recording interval. Statistics are manually rest by issuing the type-2 command UPDATE MSLINK NAME(*linkname*) START(STATISTICS) OPTION(RESET).

Adjust link buffer sizes to the size of the messages

Use the MSC logical link statistics to determine what the messages sizes are for an MSC link and adjust the buffer sizes for the link and message queues so that most messages can be received with a single I/O operation and two QGET (one GU and one DEQ) calls.

The high, low, and average message size statistics returned by the type-2 command QUERY MSLINK are derived from the total sizes of the messages in the send and receive message queues, including the prefix data and user data segments, as they are logged in the X'01' and X'03' log records.

Adjust the buffer sizes so that IMS uses no more than a single I/O operation and two QMGR calls to either send or receive a message on an MSC link: two QGET calls (one GU and one DEQ) and two QPUT calls (one ISRT and one ENQ).

The QPUT calls on the sending side of an MSC link operate differently from the QGET calls on the receiving side of an MSC link. Each QPUT call is for a message segment, and each QGET GU call is for a message queue buffer. A message with two user segments in one queue buffer requires three QPUT calls (two ISRT calls and one ENQ call) and two QGET calls (one GU call and one DEQ call). For single segment messages, IMS uses the same number of QPUT and QGET calls on each side of an MSC link.

Adjust logical link capacity for MSC bandwidth mode

In MSC bandwidth mode, IMS can process multiple messages in single send or receive action. When IMS processes multiple messages at a time, the total number of sends and receives is less than the total number of individual messages sent and received.

Although processing multiple messages in a single send or receive action is good for processing efficiency, if a link buffer that is sized for multiple messages is always full when the messages are sent, the link might be operating at or near its bandwidth capacity and message might be stacking up while waiting for the buffer to be cleared again.

If your logical links are operating at or near their bandwidth capacity, you might be able to ease the load on the link by distributing the message workload across additional logical links.

Determining optimum MSC link buffer sizes

The primary way to determine the optimum size of a link buffer is to use MSC link statistics to compare the number of times buffer content is sent with the number of actual messages sent.

If you are using bandwidth mode, the number of messages sent should be equal to or greater than the number of buffers sent. A buffer size that is optimized for bandwidth mode allows for multiple messages to be sent in a single send of the buffer contents, if there are multiple messages on the output queue.

If you are using non-bandwidth mode, the number of messages sent should be approximately equal to the number of buffers sent. A buffer size that is optimized for non-bandwidth mode allows for one complete message to be sent in each send of the buffer content.

To determine how well your buffer size is set, issue the command **QUERY MSLINK** NAME(*linkname*) SHOW(STATISTICS) and compare Tot_Send_CT, which is the total number of buffers sends, and Tot_Msg_Send_CT, which is the total number messages sent.

If you are using bandwidth mode, the Tot_Msg_Send_CT value should be equal to or greater than the Tot_Send_CT value. If the Tot_Msg_Send_CT is greater than the Tot_Send_CT value, multiple messages are being sent in each send of the buffer contents. If not, the traffic rate in the link might not be

high enough to fill the buffer and the Tot_Send_CT value will be greater to reflect the acknowledgment messages sent by IMS.

If you are using non-bandwidth mode, the Tot_Msg_Send_CT value should be approximately equal to the Tot_Send_CT value, which indicates that each send of the buffer contents contains a complete message. If the Tot_Msg_Send_CT value is less than the Tot_Send_CT value, the buffer size is not large enough to contain a complete message and multiple buffer sends are required to send a complete message.

Related reference

QUERY MSLINK command (Commands)

Determining MSC link buffer sizes for bandwidth mode

When MSC links run in bandwidth mode, you need to account for some additional bytes when determining the size of your link buffers.

In bandwidth mode each send of the buffer contents can include multiple responses and messages. However, if nothing else is on the output queue, a send of the buffer contents can include only one response or one message.

Buffer work fields are not included in the contents sent from the buffer.

Table 137. Byte format of MSC BUFMS buffers in bandwidth mode

Link type	Buffer work fields	Buffer header	Each response	Each message	VTAM work area
CTC	96	128	96	Size of message data, as shown in the X'01' or X'03' log record, plus 144-byte bandwidth prefix	N/A
MTM	96	128	96	Size of message data, as shown in the X'01' or X'03' log record, plus 144-byte bandwidth prefix	N/A
TCP/IP	96	240	96	Size of message data, as shown in the X'01' or X'03' log record, plus 240-byte bandwidth prefix	N/A
VTAM	96	128	96	Size of message data, as shown in the X'01' or X'03' log record, plus 144-byte bandwidth prefix	196

To calculate buffer sizes to use with MSC links in bandwidth mode:

1. Include 96-bytes for the buffer work field.

   The buffer work field is always the first 96 bytes of each buffer. The buffer work field is not sent.
2. Include the bytes for the buffer header that is in the front of each buffer sent. The size of the buffer header is determined by your link type:
   - For CTC links, include 128 bytes for the buffer header.
   - For MTM links, include 128 bytes for the buffer header.
   - For TCP/IP links, include 240 bytes for the buffer header.
   - For VTAM links, include 128 bytes for the buffer header.
3. For VTAM links only, include 196 bytes for the VTAM work area.
4. Include 96 bytes for responses.

   Each response is 96 bytes. The number of responses depends on the number of messages sent. Responses are inserted into the buffer first.
5. Determine the size of the messages being sent.

   The size includes both the message prefix, data, and a 144-byte bandwidth prefix.

You can check the sizes of messages sent by issuing the **QUERY MSLINK** NAME(*linkname*) SHOW(STATISTICS) and look at the Hi_Msg_Send_SZ and Avg_Msg_Send_SZ values. The sizes returned by the QUERY MSLINK command include only the message prefix and data, but not the bandwidth prefix. The sizes returned are the same as the type X'01' and type X'03' queue manager message sizes.

The Hi_Msg_Send_SZ and Avg_Msg_Send_SZ values are derived from the actual sizes of the messages sent, as recorded in the X'01' and X'03' log records. The sizes in X'01' and X'03' log records include the prefix data and user data segments of each message. The X'01' and X'03' log records are mapped by the QLOGMSGP macro.

Related reference

QUERY MSLINK command (Commands)

Log records (Diagnosis)

Determining MSC link buffer sizes for non-bandwidth mode

In non-bandwidth mode, you do not need to account for responses when determining the size of MSC link buffers.

You also do not need to account for the bandwidth header that is sent with each message when bandwidth mode is used.

In non-bandwidth mode, each send of the buffer contents includes only a single message. Responses are not sent as separate objects in the buffer contents.

Buffer work fields are not included in the contents sent from the buffer.

The TCP/IP link type does not support non-bandwidth mode.

Table 138. Byte format of MSC BUFMS buffers in non-bandwidth mode

Link type	Buffer work fields	Buffer header	Response	Message	VTAM work area
CTC	96	6	Not applicable	Size of message data, as shown in the 01 or 03 log record, minus 124 bytes	Not applicable
MTM	96	6	Not applicable	Size of message data as shown in the 01 or 03 log record minus 124 bytes	Not applicable
TCP/IP	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable
VTAM	96	23	Not applicable	Size of message data as shown in the 01 or 03 log record minus 124 bytes	196

To calculate buffer sizes to use with MSC links in non-bandwidth mode:

1. Include 96-bytes for the buffer work field.

   The buffer work field is always the first 96 bytes of each buffer. The buffer work field is not sent.
2. Include the bytes for the buffer header as determined by your link type:
   - For CTC links, include 6 bytes for the buffer header.
   - For MTM links, include 6 bytes for the buffer header.
   - For VTAM links, include 23 bytes for the buffer header.
3. For VTAM links only, include 196 bytes for the VTAM work area.

   The data in the VTAM work area is not sent with the messages.
4. Calculate and include the size of the messages that are sent.

In non-bandwidth mode, the size of the message that is sent is the message size shown in the X'01' and X'03' log records minus 124 bytes. In non-bandwidth mode, certain message prefixes are not sent with the message data and instead are rebuilt by the receiving IMS system.

You can check the sizes of messages sent by issuing the **QUERY MSLINK** NAME(*linkname*) SHOW(STATISTICS) and look at the Hi_Msg_Send_SZ and Avg_Msg_Send_SZ values.

The Hi_Msg_Send_SZ and Avg_Msg_Send_SZ values are derived from the actual sizes of the messages sent by subtracting 124 bytes from the recorded size in the X'01' and X'03' log records. The sizes in X'01' and X'03' log records include the prefix data and user data segments of each message. The X'01' and X'03' log records are mapped by the QLOGMSGP macro.

Related reference

QUERY MSLINK command (Commands)

Log records (Diagnosis)

Use high-value link statistics to help diagnose MSC link problems

You can use high-value link statistics to help diagnose certain problems that lead to a backup of messages on MSC links. In the output of the type-2 command QUERY MSLINK command, the fields for the high-value statistics all begin with HI.

For example, if a message backup on a link occurs, you might check for an unusually high value recorded for the Hi_SendIO_Time. An unusually high I/O time to send a message might indicate that an outage has occurred in the network connecting the two IMS systems and you can then pursue the cause.

The high-value statistics that you can check include:

Hi_Proc_Time
The highest (longest) time the link was dispatched to process

Hi_MSG_Send_SZ
The largest message size sent (type X'01' or X'03' message record)

Hi_QGET_Time
The highest (longest) QMGR call (GU or DEQ) to process a send message.

Hi_SendIO_Time
The highest (longest) I/O time to send a message

Hi_MSG_Rec_SZ
The largest message size received (type X'01' or X'03' message record)

Hi_QPUT_Time
The highest (longest) QMGR call (ISRT or ENQ) to process a received message

Hi_RecIO_Time
The highest (longest) I/O time to receive a message

Monitoring and tuning multiple systems

Plan to obtain both statistical and performance data for IMS online systems that are part of a MSC network.

You can use the same monitoring tools that are used for generating performance data for single IMS systems:

- You can execute the IMS Monitor in several systems concurrently. You obtain IMS Monitor reports for each individual IMS system and coordinate your processing analysis.
- The Statistical Analysis utility produces summaries of transaction traffic for each individual system. Again, you combine the statistics for a composite picture.
- The IMS Transaction Analysis utility enables you to trace transactions across multiple systems and examine the traffic using the various active physical links.

Coordinating performance information

If possible, expand your MSC network by increasing the number of SYSIDs and the number of physical links and logical links.

You can specify up to 2036 SYSIDs. You can define up to 999 physical links and 999 logical links.

By expanding the MSC network, you can:

- Access an IMS subsystem from many other IMS subsystems
- Route transactions
- Distribute transaction processing
- Increase network throughput
- Grow beyond the capacity of one IMS system
- Respond to capacity constraints or response-time constraints

The IMS system log for each system participating in MSC contains only the record of events that take place in that system. However, logging is performed to record traffic on the links. Add the SYSIDs of all coupled systems to the system log documentation that records the checkpoint intervals. This helps to interpret reports, because you are aware of transactions that might be present in message queues but are not processed, and you can expect additional transaction loads from remote sources.

Your analysis procedures should include ways of isolating the processing that is triggered by transactions originating from another system. Considerations for tuning buffers for the asynchronous communication processing should include a criterion that no exceptional conditions resulting from intersystem traffic exist.

To satisfy the need for monitoring with typical activity that includes cross-system processing, coordinate your scheduling of the DC Monitor and other traces between master terminal operators. The span of the monitoring does not need to be exactly the same, but if it is widely different, the averaging of report summaries might make it more difficult to interpret the effect of the processing that is triggered by cross-system messages.

Reports generated by the IMS Monitor for MSC

The IMS Monitor Report Print Program includes three reports that highlight message events caused by system coupling.

The reports are:

MSC Traffic report
: Shows for a specified interval, the counts of messages using the various link paths. This report can be used to assess cross-system queueing

MSC Summaries report
: Shows summaries of the traffic queues for each input transaction name, each destination name, each link number, and each destination system. The MSC Summaries report can be used to assess link loads.

MSC Queuing Summary report
: Shows how long messages spend on queues and the numbers of messages on queues. The MSC Queueing Summary Report is generated when intersystem messages are queued on the local system before the messages are sent to the destination system. The local system must be an intermediate system.

All three reports can have entries in the Distribution Appendix, so that you can examine the frequency distributions of the traffic if you suspect unusual transmission patterns. The Distribution Appendix displays multiple occurrences of an event across a range of values relevant to the event, such as wait times. For example, in the case of wait times, the Distribution Appendix allows you to easily see how often a given event occurs with a specific wait time and what the distribution of the event is across a range of wait times.

Related concepts

IMS Monitor reports (System Administration)
MSC Traffic report (System Administration)
MSC Summaries report (System Administration)
MSC Queuing Summary report (System Administration)
Interpreting the Distribution Appendix (System Administration)

Extracting multiple-system transaction statistics

You can use the Log Transaction Analysis utility to obtain counts of the message traffic both in local systems and between systems.

The transmissions over the different types of physical links can also be examined. (The activity is summarized for each step of the logical link paths.) You must provide IMS system log input that reflects all partner system activity, that is, sets of system logs for each MSC system. To coordinate the sets of individual system logs, use the Log Merge utility. As many as nine separate system logs can be merged, each log being the output of a uniquely identified IMS system with MSC installed.

Related reference

Log Transaction Analysis utility (DFSILTA0) (System Utilities)

Controlling the log merge

You need to perform a number of steps to control log output.

To control the log output, you need to:

- Choose the required systems that participate in the logical link paths you want to examine.
- Coordinate the series of input logs for each system so that they cover a similar time span.
- Specify a start and stop time for the Log Merge utility control statements if you want to sample the cross-system processing for a particular interval.

  You can give both start date (Julian) and time of day, or just time of day. It is the first system log (specified by the LOG01 DD statement) to which these times apply. Other log activity is collected if it falls between the initial and final events that are present on the first log.
- Specify MSG to select log records that are suitable for the transaction analysis step. (ALL records is the default, but this means the DL/I activity for several systems is included in the utility input, and this can cause extended processing time.)

Related reference

Log Transaction Analysis utility (DFSILTA0) (System Utilities)
Log Merge utility (DFSLTMG0) (System Utilities)

Interpreting the Transaction Analysis report

Using the Log Analysis report produced by the IMS Transaction Analysis utility, you can obtain statistics for individual transactions that are processed in any system.

The Log Analysis report includes the following statistics:

The total response time
The time on input and output queues
The processing time

The Log Analysis report also includes definitions for the format of the detailed report records produced by the IMS Transaction Analysis utility and a list of processing type codes. The absence of times for a message GU call or MPP termination in the report lines indicates an input source or intermediate system report line.

The processing type field is an important one for the interpretation of the detailed report lines. The S code indicates that this line shows a send or receive event for the transaction. You can trace the progress of a cross-system conversation using the codes C, D, P, X, and Y.

The report headings include a column headed ID after the column for the GU to the message queue time. The number shown in a report line under the ID heading matches the sequence in which log input is fed to the Log Merge utility. The field corresponds to starting position 102, the 3-digit field named SYSTEM ID, in the detailed report records.

You can use the sort step to reorder the report records in any order you want, for example, by system ID within transaction code. The default order is the input sequence.

Related concepts

Statistical-analysis, log-transaction reports, and analyzing log records (System Administration)

Related reference

Log Transaction Analysis utility (DFSILTA0) (System Utilities)

MSC and IMSplexes with shared queues

The following topics discuss the coexistence of a Multiple Systems Coupling (MSC) network with an IMSplex with shared queues. MSC and IMSplex coexistence can be temporary, such as when migrating from an MSC network to an IMSplex configuration, or permanent, such as when an MSC link connects an IMSplex to an IMS system outside of the IMSplex.

A primary concern when an MSC network and an IMSplex coexist is the proper routing and processing of your transaction messages across both the MSC and IMSplex environments, because each environment uses a different routing method.

Generally, MSC networks route transactions to specific IMS systems using SYSIDs, while IMSplexes with shared queues route transactions by making them available on the shared queue to any IMS system that registers an interest in the transactions. When an MSC network and an IMSplex with shared queues coexist, both of these methods of routing can apply to transactions.

Message routing across MSC and IMSplex environments

MSC uses local and remote SYSIDs and destination names to route messages across links between local, intermediate, and remote IMS systems in an MSC network. IMSplexes with shared queues use destination name registration, IMSIDs, a shared queue, and the registered interest of IMS systems to route messages between front-end and back-end IMS systems in an IMSplex. When an IMSplex and an MSC network coexist, both methods of routing can be used.

In MSC networks and IMSplexes with shared queues, IMS stores the SYSID, destination name, and IMSID value in the message prefix when IMS builds the message and places it on a local or shared queue. The SYSID, destination name, or IMSID stored in the message prefix must match the SYSID, destination name, or IMSID of an IMS system in either the IMSplex or the MSC network. If they do not match, any messages on the shared queue or in flight in the MSC network will trigger a routing error, such as a pseudoabend U0830, when an IMS system attempts to process the message. This condition can occur, for example, if the IMS system that owns the SYSID, destination name, or IMSID is down or has changed its value.

How messages are routed in an MSC network

In an MSC network, IMS uses remote transactions, remote logical terminal (LTERM) names, and SYSIDs to route the messages through the MSC network.

Remote transactions are defined either dynamically with the CREATE TRAN command or statically by specifying remote and local SYSIDs in the TRANSACT macro. Remote LTERMs are defined either dynamically by the CREATE LTERM command, or statically with the NAME macros immediately following an MSNAME macro in the stage-1 system definition.

MSNAME labels can also be used as intermediate destination names, when messages must pass through an intermediate IMS system prior to reaching the IMS system that contains the true destination name. In this case, the MSNAME label is also stored in the message prefix.

Related concepts

"Routing messages with the destination name and SYSIDs" on page 680
Messages in an MSC network contain information that makes it possible to route the message between IMS systems.

Related tasks

"Defining Multiple Systems Coupling resources" on page 689
Multiple Systems Coupling (MSC) resources, such as physical links, logical links, and system identifiers, can be defined either dynamically while IMS is running (using IMS type-2 commands) or statically during IMS system definition (using macros).

Related reference

MSNAME macro (System Definition)
NAME macro (System Definition)
TRANSACT macro (System Definition)
CREATE TRAN command (Commands)
CREATE LTERM command (Commands)

How messages are routed in an IMSplex with shared queues

In an IMSplex with shared queues, IMS uses origin names and destination names registered with a coupling facility to route messages between front end and back end IMS systems.

Origin and destination names can be logical terminal names, transaction codes, or APPC or OTMA client names. You define static origin and destination names during system definition. You define in descriptor libraries or user exits the origin and destination names that are dynamic LTERMs, ETO terminals, or dynamic transactions. Origin and destination names for APPC or OTMA clients are sent, in the form of an 8-byte token, to IMS when an APPC or OTMA client allocates a conversation.

IMS registers transactions when they are started and a dependent region or an MSC link is started and ready to process messages. IMS registers LTERM names with the coupling facility when they are started or when they sign on and are ready to process messages. IMS registers APPC and OTMA tokens when an APPC or OTMA conversation is allocated. IMS deregisters APPC and OTMA tokens when the APPC or OTMA conversation is deallocated.

Messages carry these origin names and destination names in the message prefix. When an empty shared queue on a coupling facility receives a message from a front-end IMS system for a registered destination name, the coupling facility notifies all interested IMS systems that there are messages to process.

Message routing when an IMSplex and MSC network coexist

When an IMSplex with shared queues and an MSC network coexist, IMS routes messages between front-end, back-end, and remote IMS systems using origin and destination names, MSC SYSIDs, and IMSIDs.

IMS uses the coupling facility and the shared queues to route messages within the IMSplex. IMS uses MSC links only to route messages to IMS systems outside of the IMSplex. Any MSC link definitions that exist between IMS systems in the IMSplex are not used.

Transactions defined for routing in an MSC network can also be routed in an IMSplex with shared queues but the method used for routing is that of the IMSplex and not MSC; however, the IMSplex does recognize the MSC SYSID attributes of the transaction. The recognition of the SYSIDs by the IMSplex allows the IMSplex to handle remote transaction in a manner similar to the MSC network. This also has implications for local processing affinity.

Transactions not specifically defined for routing in an MSC network cannot be routed outside of the IMSplex using an MSC link.

Related concepts

"Processing affinities in an IMSplex" on page 715
If a message must process on a specific IMS system in the IMSplex, IMS assigns the message affinity to that IMS system by appending the IMSID of the IMS system to the destination name. Then, only that IMS can process it.

Processing affinities in an IMSplex

If a message must process on a specific IMS system in the IMSplex, IMS assigns the message affinity to that IMS system by appending the IMSID of the IMS system to the destination name. Then, only that IMS can process it.

You can also use the TM and MSC Message Routing and Control user exit routine (DFSMSCE0) to establish affinities between transaction instances and IMS systems in an IMSplex.

Processing affinities in an IMSplex-MSC coexistent configuration

In an MSC network, remote transactions bypass any local affinity the transactions might have to the IMS system that receives them as input. In an IMSplex, remote transactions also cause local affinity to be bypassed. Otherwise, the rules that govern processing affinities in an IMSplex remain unchanged.

Related concepts

"TM and MSC Message Routing and Control user exit routine overview" on page 730
Message routing is automatic, according to the defined scheme, unless you use the TM and MSC Message Routing and Control user exit routine (DFSMSCE0), which provides routing options and control of messages.

Related tasks

"Managing remote transactions for APPC and OTMA when MSC and IMSplexes coexist" on page 719
The following topics provide information about migrating from an MSC network to an IMSplex when synchronous APPC and OTMA messages were processed remotely in the MSC network, and processing APPC and OTMA messages on a remote IMS system outside of the IMSplex when an IMSplex and an MSC network coexist.

Migrating from an MSC network to an IMSplex network

When migrating or converting an existing MSC network to an IMSplex with shared queues, you need to retain the MSNAME definitions in your IMS systems for as long as you have transaction messages that use those definitions on the shared queues.

You also need to retain any MSC link definitions to IMS systems outside the IMSplex. Once you have finished testing the IMSplex and are sure you will not need to return any IMS systems in the IMSplex to an MSC-only network, you can remove any MSC link definitions between IMSplex member systems.

In an IMSplex, if one IMS system is MSC capable, then all the IMS systems in the IMSplex must be MSC capable. To enable an IMS system for MSC, define an MSC link for the IMS system during system definition. This MSC link definition does not need to be functional.

MSC link definitions in an IMSplex

During the migration process from an MSC network to an IMSplex, you can retain your MSC link definitions for as long as you have a need for them. This allows you to test the IMSplex with shared queues environment and then revert to the MSC network to make adjustments.

After you have established the IMSplex with shared queues, you should not add new MSC links between the IMS systems in the IMSplex. If you attempt to start an MSC link between two IMS systems within the same IMSplex with shared queues, IMS issues message DFS2149 and aborts the link startup.

Sharing MSNAME definitions and SYSIDs in an IMSplex

In an IMSplex with shared queues, when you start an IMS system that includes MSNAME statements, the IMSplex shares the MSNAME statements and the local SYSIDs of the joining IMS system with the other IMS systems already in the IMSplex.

For each MSNAME statement in the joining IMS system, the IMSplex creates a duplicate dynamic MSNAME statement in each of the other IMS systems in the IMSplex, unless one already exists.

For each local SYSID owned by the joining IMS system, the IMSplex adds a duplicate local SYSID to the SYSID table in each of the other IMS systems in the IMSplex. This effectively makes the SYSID of the joining IMS system local to the IMSplex as a whole. The IMSplex updates the SYSID table of each IMS system in the IMSplex whenever an IMS system that has an MSNAME statement joins or rejoins the IMSplex.

The generation of dynamic MSNAME statements and the sharing of local SYSIDs throughout the IMSplex allow the IMSplex to function as a single IMS system on the MSC network. It also allows transactions defined to run in an MSC network to take advantage of the distributed processing of the IMSplex with shared queues.

To illustrate dynamic MSNAMEs and shared SYSIDs, The following figure shows a simple MSC network prior to introducing an IMSplex with share queues. Figure 123 on page 717then shows the same MSC network after an IMSplex with shared queues has been created between two of the IMS systems.

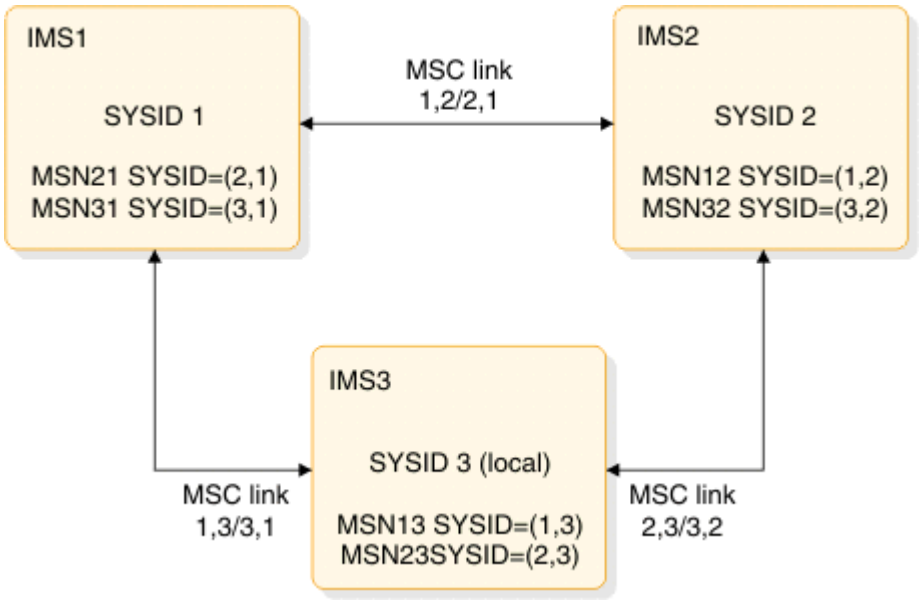


Figure 122. MSC network without an IMSplex

In the previous figure, IMS1, IMS2, and IMS3 are each members of the MSC network. Each of their local SYSIDs are unique and their MSNAME statements only define the links that are defined locally in each IMS system.

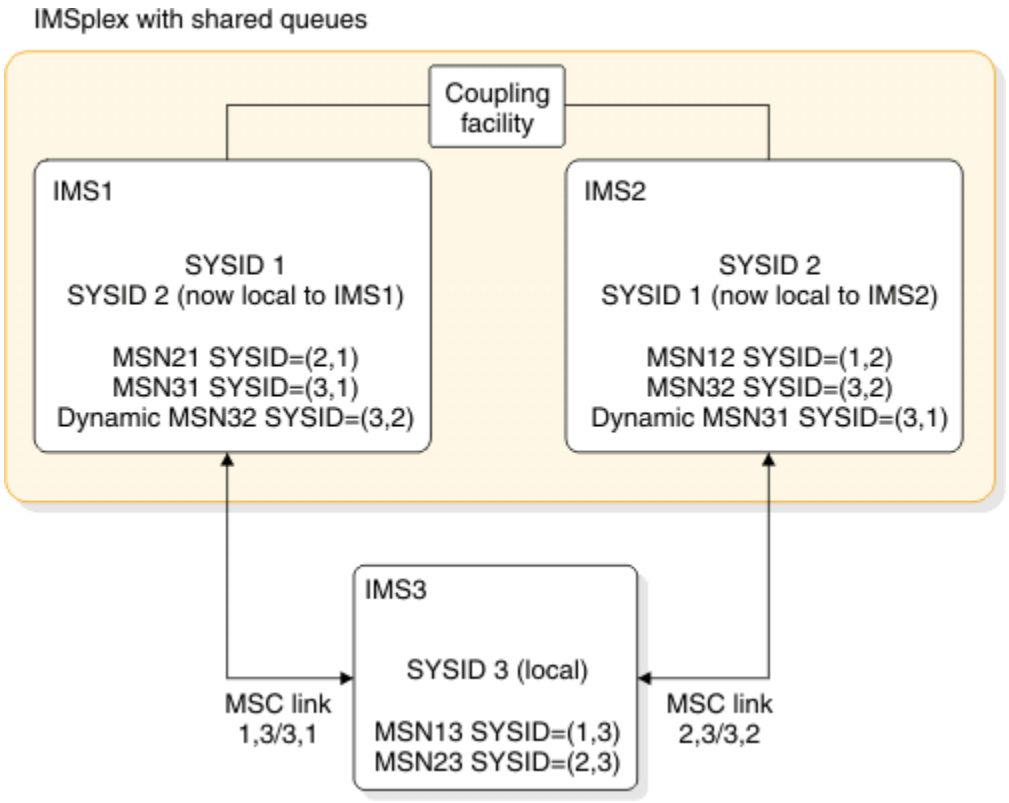


Figure 123. MSC network coexisting with an IMSplex with shared queues

In the preceding figure, IMS1 and IMS2 are now part of an IMSplex with shared queues. Because of this, SYSID 1 becomes local to IMS2 and SYSID 2 becomes local to IMS1. Similarly, dynamic MSNAME MSN31 appears in IMS2, and dynamic MSNAME MSN32 appears in IMS1. IMS1 and IMS2 also remain connected to the MSC network and IMS3, although IMS1 and IMS2 now function as a single IMS system, or node, within the MSC network.

The following table represents the SYSID tables of the IMS systems shown in Figure 122 on page 716 prior to introducing the IMSplex:

Table 139. SYSID ownership in an MSC network without an IMSplex

SYSID	IMS1	IMS2	IMS3
1	Local	RMT/MSN12	RMT/MSN13
2	RMT/MSN21	Local	RMT/MSN23
3	RMT/MSN31	RMT/MSN32	Local

The following table represents the resulting SYSID tables in each IMS after the IMSplex is introduced into the MSC network, as shown in Figure 123 on page 717:

Table 140. SYSID ownership in an MSC network that coexists with an IMSplex

SYSID	IMS1	IMS2	IMS3
1	Local	Local	RMT/MSN13
2	Local	Local	RMT/MSN23
3	RMT/MSN31 Dynamic MSN32	RMT/MSN32 Dynamic MSN31	Local

When an IMS system leaves the IMSplex, the IMSplex does not change the SYSID tables or delete any dynamic MSNAME statements in the remaining IMS systems. This ensures that if there are any messages on the shared queues that use the MSNAME definitions of the departed IMS, the IMSplex can still process them. When an IMS system rejoins the IMSplex, the IMSplex always exchanges MSNAME statements again and revalidates the rejoining SYSID on all SYSID tables in the IMSplex.

Whenever an IMS system successfully joins or leaves an IMSplex with shared queues, the IMSplex issues message DFS0778I to the master terminal of each IMS system in the IMSplex.

MSNAME duplication in an IMSplex with shared queues

For each IMS system that participates in both an MSC network and a shared queues group, MSNAMEs and their associated remote and local SYSIDs are exchanged with the other IMS systems in the shared queues group when the IMS is initialized.

Dynamic MSNAMEs are created in each IMS system for the MSNAMEs that were defined on the other IMS systems in the shared queues group.

A common MSC SYSID routing table is built in each IMS system from the remote and local SYSID values that are defined in each MSNAME. SYSIDs that are local in each IMS system are also made local to the other IMS systems in the shared queues group. The same local SYSIDs on all IMS systems in the shared queues group allow any message received from an IMS system outside the shared queues group through an MSC link to be routed and processed by any IMS system in the shared queues group.

Duplicate MSNAMEs in IMS systems in a shared queues group must have the same remote SYSID because MSNAMEs and remote SYSIDs are synonymous. The same MSNAMEs must relate to the same remote SYSIDs. Different local SYSIDs for the same MSNAME are allowed.

If the same MSNAME is defined with different remote SYSIDs in two IMS systems, the MSNAME is ignored when MSNAMEs are exchanged within the shared queues group. Dynamic MSNAMEs are not created in the IMS systems with the same MSNAME and each IMS retains its own version of the MSNAME.

Each of these two examples show valid combinations of MSNAMEs that have the same name in a shared queues group:

- A valid MSNAME pair:
  - (On IMSA) `MSN12345 MSNAME SYSID=(1,2)`
  - (On IMSB) `MSN12345 MSNAME SYSID=(1,2)`
- Also a valid MSNAME pair:
  - (On IMSA) `MSN12345 MSNAME SYSID=(1,2)`
  - (On IMSB) `MSN12345 MSNAME SYSID=(1,3)`

This is an example of an invalid combination of MSNAMEs that have the same name in a shared queues group:

- (On IMSA) `MSN12345 MSNAME SYSID=(1,2)`
- (On IMSB) `MSN12345 MSNAME SYSID=(2,1)`

Related concepts

Resource type consistency (System Administration)

Deleting MSNAME definitions from the SYSID tables in an IMSplex

To remove a SYSID or an MSNAME statement from the IMSplex, you must first remove it from the IMS system that owns it and then cold start the IMS system back into the IMSplex.

After you have removed the SYSID or MSNAME statement from its original owning IMS system, warm or cold start each of the other IMS systems in the IMSplex to rebuild their SYSID tables.

Removing MSPLINK and MSLINK definitions when an MSC to IMSplex migration is complete

After the migration of an MSC network to an IMSplex with shared queues is complete and you are sure you will not return any IMS systems to their MSC-only state, you should remove the MSPLINK and MSLINK definitions for the MSC links that you will no longer use.

Note: You must retain at least one MSC link definition in each IMS system in the IMSplex to maintain MSC enablement throughout the IMSplex. This link can be a functioning MSC link to an IMS system outside of the IMSplex or a non-functioning link defined only to attach MSNAME statements to and to ensure MSC enablement.

Managing SYSIDs when MSC and IMSplexes coexist

Local SYSIDs must be unique in an MSC network. Local SYSIDs are not required to be unique within an IMSplex; however, local SYSIDs owned by IMSplex members linked to an MSC network outside of the IMSplex must be unique in that MSC network.

Also, if multiple members of an IMSplex link to the same IMS system outside of the IMSplex, each IMSplex member must use a local SYSID that is unique to that outside IMS system.

You cannot use the same remote SYSID to point to different IMS systems in an MSC network.

Cloning MSC SYSIDs in an IMSplex

Even though the IMSplex adds every local SYSID in the IMSplex to each SYSID table in the IMSplex, you should clone all local SYSIDs within each IMS system in the IMSplex when removing the intra-IMSplex MSC links. To clone a local SYSID, clone an MSNAME statement that contains it.

Cloning local SYSIDs ensures that the IMSplex can route and process messages on a shared queue even if the IMS system that owns the SYSID is not available and the SYSID is not included in the SYSID tables in the IMSplex. MSC SYSIDs are not stored in the coupling facility within the IMSplex. If an IMS system cannot recognize the SYSID of a message on the shared queue as local when processing application program messages, the IMSplex issues pseudoabend U0830.

IMSIDs when IMSplexes and MSC coexist

The IMSplex uses IMSIDs to identify IMS systems and to route any messages that have an affinity to a specific IMS system. IMSIDs must be unique within each IMS in an IMSplex.

In an IMSplex-MSC network, IMSIDs must be unique throughout the entire MSC network, even the remote MSC IMS systems outside of the IMSplex.

IMSIDs are not required to be unique in MSC networks that do not include an IMSplex. When migrating an MSC network to an IMSplex with shared queues, make sure the IMSIDs of each IMS system are unique.

Managing remote transactions for APPC and OTMA when MSC and IMSplexes coexist

The following topics provide information about migrating from an MSC network to an IMSplex when synchronous APPC and OTMA messages were processed remotely in the MSC network, and processing APPC and OTMA messages on a remote IMS system outside of the IMSplex when an IMSplex and an MSC network coexist.

Remote processing of APPC and OTMA messages in an MSC network

In an MSC network, you can bypass local affinity for synchronous APPC and OTMA messages by queuing them to remote MSC transactions.

Remote IMS systems process APPC and OTMA messages asynchronously and disassociated with the APPC or OTMA conversation. IMS does not propagate or cascade APPC or OTMA conversations to the remote IMS system. IMS saves an APPC or OTMA token in the message prefix. When a response

or program-to-program switch returns to the originating IMS system, IMS attempts to restore the synchronous APPC or OTMA conversational mode to the message.

To enable the remote processing of an APPC or OTMA message in an MSC network, queue the message to a remote MSC transaction.

Back-end processing of APPC or OTMA transaction messages in an IMSplex with shared queues

In an IMSplex with shared queues, you can use remote MSC transactions to route APPC or OTMA messages to back-end IMS systems for processing.

Using remote MSC transactions avoids the local affinity restrictions that are otherwise imposed on transaction messages received from APPC or OTMA clients. Any IMS system in the IMSplex that defines the transaction as a local MSC transaction that is assigned to a region can then process the transaction.

If you are migrating from an MSC network to an IMSplex, you can use existing remote MSC transactions to process APPC or OTMA messages on back-end IMS systems in the IMSplex and bypass the APPC and OTMA affinity restrictions. When the transactions are converted to local IMSplex transactions, the APPC and OTMA affinity restrictions still apply unless you specify RRS=Y and AOS=Y.

When an APPC or OTMA message is queued in a front-end IMS system to a transaction that is defined as a remote MSC transaction, IMS inserts the transaction message to the shared queue without affinity to any IMS system.

When a back-end IMS system retrieves APPC and OTMA transaction messages from the shared queue, it saves an APPC or OTMA conversation token in the message prefix and processes the transaction message independently from, and asynchronous to, the APPC or OTMA conversation. IMS does not propagate or cascade the APPC or OTMA conversation to the back-end IMS system or to any other IMS systems, including the front-end IMS system, that might process APPC or OTMA messages after a program-to-program switch.

Note: If an OTMA transaction that is processing independently from the OTMA conversation issues a CHNG call to a modifiable PCB and IMS calls the OTMA Destination Resolution user exit (OTMAYPRX), the OTMAYPRX user exit recognizes the transaction message as an OTMA message only if the transaction is processing within the IMSplex where the OTMA client originally submitted the transaction.

When the originating front-end IMS system receives the first or only response, the response is returned to the client in APPC or OTMA synchronous mode, assuming that the client is still connected in synchronous mode. IMS returns any subsequent responses to the client for the same interaction asynchronously. If the client has terminated before the first or only response is returned, the response is not discarded, but instead queued to the client asynchronously.

For example, if you define an MSC transaction as remote in a front-end IMS system without assigning the transaction to a started MSC link and then you define the same MSC transaction as local in the back-end IMS and assign the transaction to a started region, when an APPC or OTMA client initiates the transaction on the front-end IMS system in synchronous mode, the front-end IMS system saves a token in the message prefix to identify the client and places the transaction on the shared queue without any affinity to the front-end IMS system.

When the back-end IMS system retrieves the transaction from the shared queue, the transaction is processed asynchronously and disassociated from the APPC/OTMA client.

The following series of figures show various possible scenarios in which remote MSC transactions can be used to process APPC or OTMA transactions on the back end in an IMSplex, as well as outside of an IMSplex by using an MSC link.

The following figure shows a simple configuration in which a remote MSC transaction is used to process an APPC or OTMA transaction on the back end in an IMSplex. The transaction TRAN1 is defined to IMS 1 as a remote MSC transaction, but because TRAN1 is not assigned to an MSC link in IMS 1, IMS 1 queues it to the shared queue.

After submitting the transaction to IMS 1, the APPC or OTMA client waits in a receive state for the synchronous response; however, the queuing and processing of TRAN1 by IMS 1 and IMS 2 is handled by IMS independently from the synchronous communications mode that is maintained with the client.

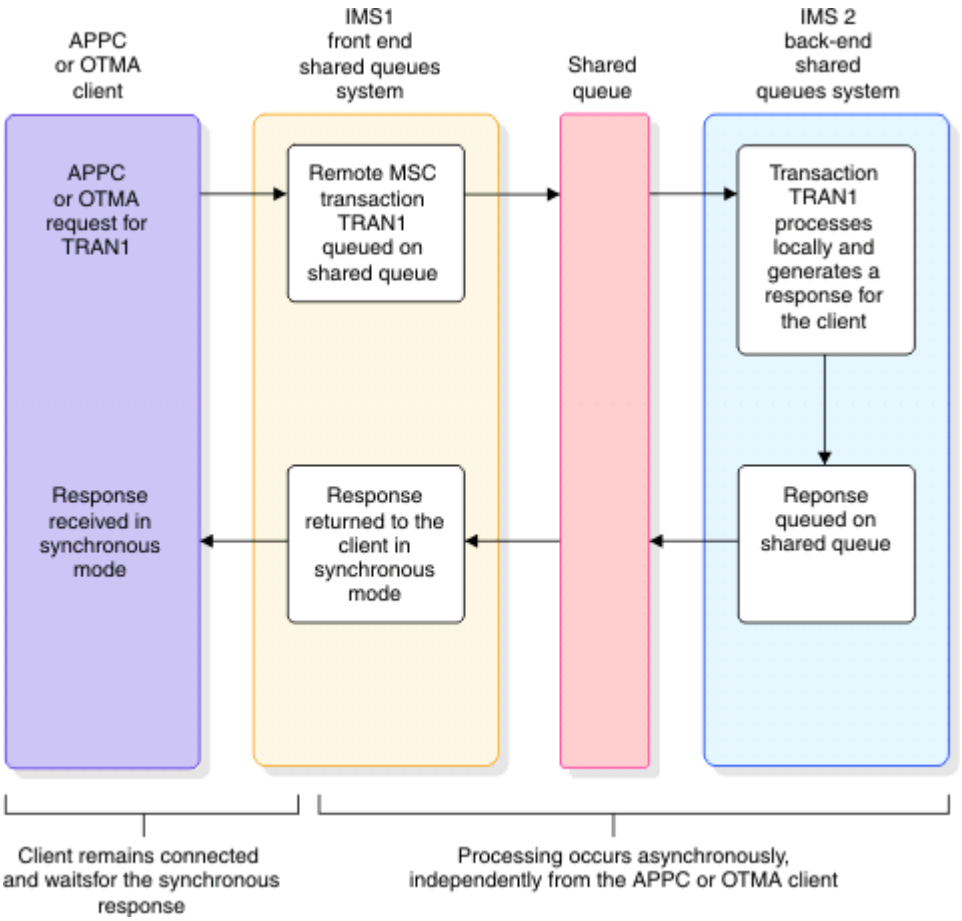


Figure 124. Using a remote MSC transaction to route an APPC or OTMA transactions to a back-end IMS system

The following figure shows another scenario in which a remote MSC transaction is used to route APPC or OTMA transaction. This scenario involves three IMS systems and TRAN1 is routed to a remote MSC IMS system by way of both a shared queue and an MSC link.

When IMS 1 receives the transaction from the client, IMS 1 queues it to the shared queue. In IMS 2 the transaction TRAN 1 is defined as a remote MSC transaction and assigned to the MSC link to IMS 3. IMS 3 generates the response, which is then returned across the MSC link to IMS 2, and then returned to IMS 1 by way of the shared queue. IMS 1 returns the response to the APPC or OTMA client in the synchronous mode expected by the client.

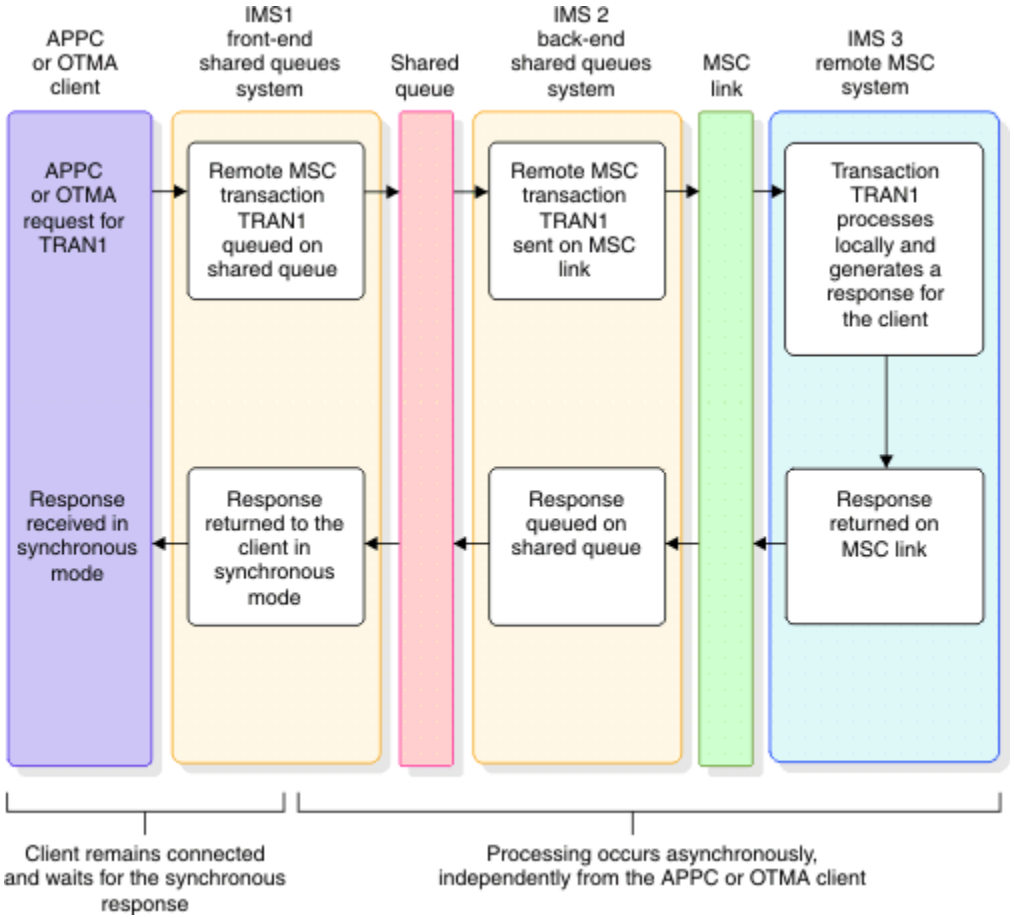


Figure 125. Routing an APPC or OTMA transaction across a shared queue and an MSC link

In the scenario shown in the following figure, the APPC or OTMA transaction TRAN1 is routed to IMS 2, a back-end IMS system in a shared queues group, where a program-to-program switch to TRAN2 sends processing back to IMS 1 by way of the shared queue. TRAN2 is not an MSC transaction. TRAN2 then sends processing back to IMS 2 by issuing another program-to-program switch to TRAN3, also a non-MSC transaction. TRAN3 is queued to the shared queue and retrieved by IMS 2, where TRAN3 processes and generates a response for the client. The response is returned to IMS 1 by way of the shared queue. IMS 1 returns it to the client in the synchronous mode expected by the client.

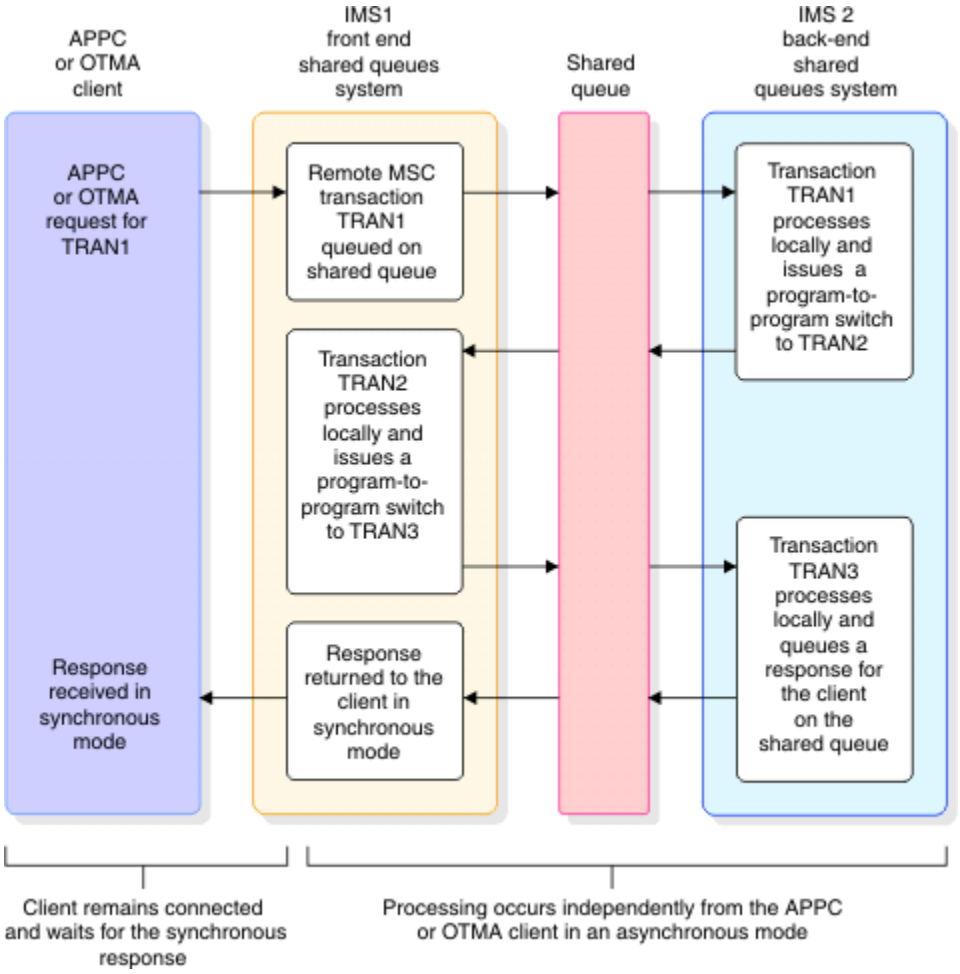


Figure 126. Routing an APPC or OTMA transaction by using a remote MSC transaction and shared queues with multiple program-to-program switches

The scenario shown in following figure involves three IMS systems. IMS 1 routes TRAN1 to IMS 2 as a remote MSC transaction by way of the shared queue. TRAN1 processes locally on IMS 2 and issues a program-to-program switch to TRAN2, which is defined as a remote MSC transaction on IMS 2. IMS 2 sends TRAN2 to IMS 3 across an MSC link. TRAN2 is processed by IMS 3 and generates the response for the client. IMS 3 returns the response to IMS 2 across the MSC link. IMS 2 queues the response to the shared queue. IMS 1 retrieves the response and returns it to the APPC or OTMA client in the synchronous mode the client expects.

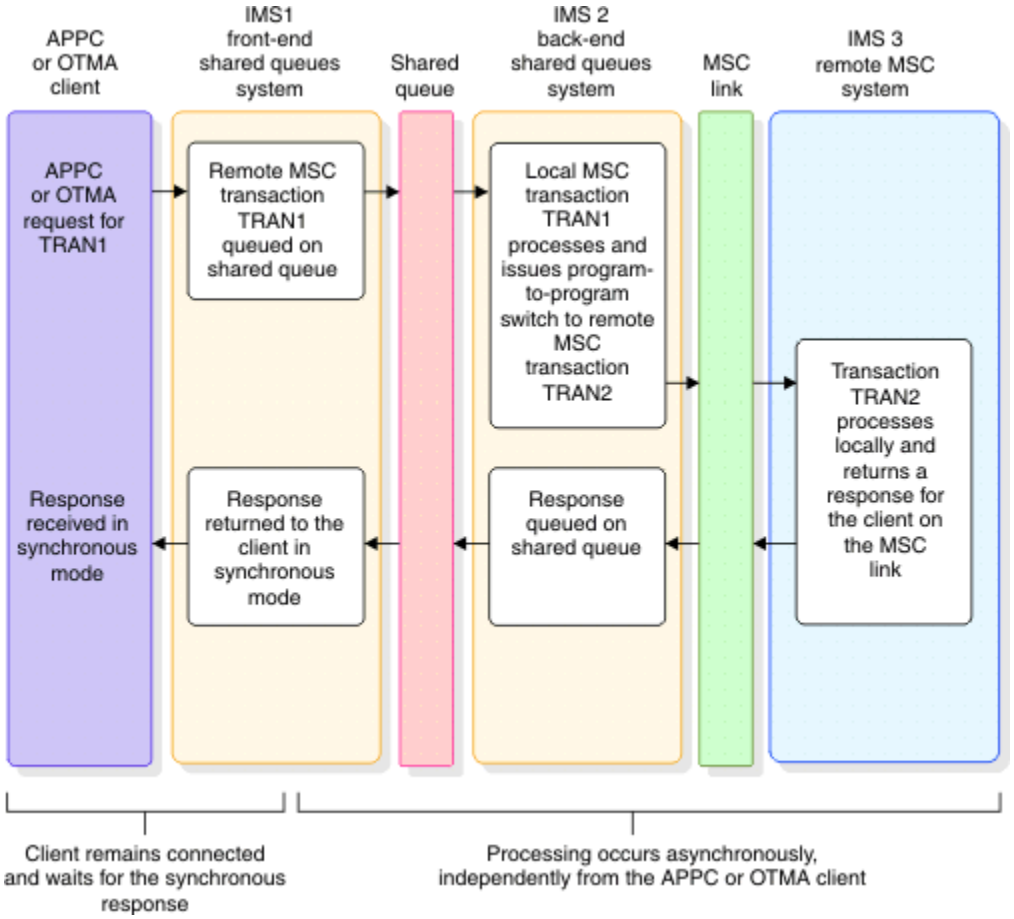


Figure 127. APPC or OTMA transaction routed using MSC and shared queues with a program-to-program switch

Related concepts

Managing APPC and OTMA messages in a sysplex environment (System Administration)

Enabling back-end processing of APPC and OTMA messages using a remote transaction

To use a remote transaction to process APPC and OTMA messages on back-end IMS systems within an IMSplex, perform the following steps.

1. Enable MSC in all IMS systems in the IMSplex by defining at least one MSC link in each IMS system in the IMSplex. This MSC link can be either a functional link to a remote IMS system outside of the IMSplex or, if none exists, it can be a non-functional MSC link.
2. In the IMS system connected to the APPC or OTMA client, define a remote transaction for the APPC or OTMA messages by specifying the SYSID keyword on the TRANSACT macro.

   In the SYSID keyword, the remote SYSID can be any remote SYSID. The local SYSID should match the SYSID of the IMS system connected to the APPC or OTMA client.
3. In all back-end IMS systems, define a local transaction to process the APPC or OTMA messages.
4. Register the remote transaction with the IMSplex by starting the transaction and the dependent regions that process it.

Sending a transaction to an MSC system outside of an IMSplex

To send an APPC or OTMA transaction from an IMSplex to a remote IMS system outside of the IMSplex by using an MSC link:

1. Enable MSC in all IMS systems in the IMSplex by defining at least one MSC link in each IMS system in the IMSplex. This MSC link can be either a functional link to a remote IMS system outside of the IMSplex or, if none exists, it can be a non-functional MSC link.
2. Define an MSC link between the IMSplex and the remote IMS system.
3. In any IMS system in the IMSplex, define a remote MSC transaction and assign it to an MSC link by specifying remote and local SYSIDs in the SYSID keyword of the TRANSACT macro.

   The remote SYSID must match the SYSID of the remote MSC IMS system. The local SYSID must match the SYSID of the IMS system that is connected to the APPC or OTMA client.
4. In the remote MSC IMS system, define a local transaction to process the APPC or OTMA transaction message.

Avoiding pseudoabend U0830

When an MSC network and an IMSplex with shared queues coexist, errors can occur if you remove IMS systems from, or add them to, the IMSplex or MSC network while messages remain on the shared queues or in flight within the MSC network.

Errors can also occur if the definitions used to route the messages, such as SYSIDs, IMSIDs, or destination names, are changed.

In an IMSplex with MSC, IMS validates SYSIDs, IMSIDs, and destination names when IMS systems retrieve transaction messages from the shared queue for processing. If validation fails, IMS leaves the transaction message on the shared queue and issues a pseudoabend U0830. By taking these precautions you can avoid most U0830 pseudoabends:

- Prior to migrating or moving IMS systems, dequeue all transaction messages destined for the IMS systems being migrated or moved from the shared queues and the MSC network.
- Prior to changing any SYSIDs, IMSIDs, or MSNAMEs, dequeue all transaction messages that use these identifiers from the shared queues and the MSC network.
- After migrating new IMS systems into an IMSplex, wait for all the IMS systems to join the IMSplex and all SYSID tables to be updated before you start dependent regions to process messages
- When restarting multiple IMS systems with MSC definitions in an IMSplex, if all local SYSIDs in the IMSplex have not been cloned in each IMS system, wait for all of the IMS systems to finish restarting before starting any dependent regions. Until restart is complete, some SYSID tables might not include all of SYSIDs that are not cloned.
- Verify that MSC is defined to each IMS system in the IMSplex. Messages queued by an IMS that does not have MSC defined will not have the MSC extension prefix and will have zero SYSID values. If this message is processed by another IMS system in the IMSplex that has the MSC feature, a pseudoabend U0830 results with return code 08.

  In an IMSplex, if one IMS system is MSC capable, then all IMS systems in the IMSplex must be MSC capable. To enable an IMS system for MSC, define an MSC link for the IMS system during system definition. This MSC link definition does not need to be functional.

For detailed information about the return codes associated with pseudoabend U0830 and other diagnosis information, see *IMS Version 15 Messages and Codes, Volume 3: IMS Abend Codes*.

MSC TCP/IP generic resources

TCP/IP generic resources enable remote MSC-enabled IMS systems to establish MSC links with a TCP/IP generic resource group instead of with specific IMS systems.

Connecting to a TCP/IP generic resource group instead of a specific IMS system makes it easier to move MSC physical links from one IMS system to another in an IMSplex without significantly impacting the remote IMS system.

For example, if the physical link partner in an IMSplex needs to be switched because of an XRF takeover or for scheduled maintenance, except for a brief stopping of the links, the switch is transparent to the IMS system outside of the IMSplex. You do not have to change the remote MSC definitions to refer to the new link partner.

IMS systems in an IMSplex participate in a TCP/IP generic resource group by specifying a shared generic IMS ID on the GENIMSID parameter of their respective DFSDCxxx member in the IMS.PROCLIB data set.

The physical links in the IMS systems on both sides of the link are defined the same way they are when TCP/IP generic resources are not used. The only difference is that the remote IMS system specifies the shared generic IMS ID on the NAME parameter of the MSPLINK system definition macro or the CREATE MSPLINK command instead of the IMS of a specific IMS system.

TCP/IP generic resources are supported in an IMSplex or XRF environment.

MSC TCP/IP generic resources work similarly to MSC VTAM generic resources (VGR), except that VTAM VGR uses a common generic APPLID name instead of a shared generic IMS ID.

When the first logical link on a physical link connects to a TCP/IP generic resource group, the logical link establishes affinity with the first IMS system in the group to accept the link request. All all subsequent logical links on the physical link also have affinity to the same IMS system.

When link attributes are displayed, logical links that are subject to affinity have a status of AFFIN. Affinity remains in effect for each logical link until all logical links to the IMS system are terminated normally. If a link terminates with an ERE status, the affinity status remains. Affinity status also remains across IMS warm starts (NRE or ERE).

IMS Connect manages the routing of links among the IMS systems that participate in a TCP/IP generic resource group. IMS Connect tracks which IMS systems have affinity with which MSC physical links.

Related tasks

Defining a TCP/IP generic resource group for MSC (System Definition)

Related reference

CREATE MSPLINK command (Commands)

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

MSC statement (System Definition)

Managing MSC links in a TCP/IP generic resource group

When MSC is used with TCP/IP generic resources, each link that connects to the TCP/IP generic resource group has affinity to a specific IMS system in the group.

Affinity restricts all of the logical link parallel sessions on a physical link to one IMS system. Affinity is cleared in IMS when a logical link is terminated normally.

When a link has affinity, a /DISPLAY AFFIN LINK or QUERY MSLINK displays the link with a status of AFFIN. The QUERY MSLINK command also shows the IMS ID of the IMS system that has affinity.

IMS Connect manages the routing of link requests to the TCP/IP generic resource group. The definitions of the MSC links in IMS Connect correlate the shared generic IMS ID of the TCP/IP generic resource group to the individual participating IMS systems in the group.

When a link to a TCP/IP generic resource group is started by the remote IMS system, the start request is routed by IMS Connect to all available IMS systems in the group. IMS Connect passes the link request to

the first IMS system to accept the link and the link affinity is set. IMS Connect then routes all logical link parallel sessions on the same physical link to that IMS system.

If a TCP/IP generic resource group contains only one IMS system, then the initial link request is always accepted by that system, regardless of which side a link is started from.

Restriction: In a TCP/IP generic resource group, where a physical link is defined in multiple IMS systems, if a logical link is in a PSTOPPED ERE state on any IMS system in the group, do not start a logical link that uses the same physical link on any other IMS system in the group. Before you can start a logical link on the same physical link on another system, you must clear the affinity of the ERE link either by shutting it down normally or by resetting it to a COLD state.

As a precaution, display the affinity status of an MSC link on its current IMS system before moving the link by restarting it on another IMS system. If the affinity is still active, shutdown the link normally, which resets the affinity status. If the link cannot be shut down normally, set the link to COLD status by issuing either the type-1 command **/CHANGE LINK** *linknum* FORCSESS | SYNCSESS COLDSESS or the type-2 command **UPDATE MSLINK**NAME(*linkname*) SET(SYNCOPT(COLDSESS)

Attention: Changing the link status from ERE to COLD or moving a link in ERE status and restarting it on another IMS system prevents the synchronization of the message sequence numbers during restart, which can cause the duplication or loss of messages.

If multiple IMS systems participate in a TCP/IP generic resource group, you can use the following methods to control which IMS system gets affinity when you start the first logical link on a physical link:

- Start the link from the IMS system in the TCP/IP generic resource group with which you need affinity to be established.
- In IMS Connect, stop the MSC physical link paths to the IMS systems in the generic resource group that do not need affinity.
  a) Issue any one of the following IMS Connect commands to stop the physical link path in IMS Connect:
     – The IMS type-2 format command **UPDATE IMSCON** TYPE(MSC) NAME*(lclPlkid*) STOP(COMM)
     – The WTOR format command **STOPMSC** *lclPlkid*
     – The z/OS MODIFY command **UPDATE MSC** NAME(*lclPlkid*) STOP(COMM)
  b) After the links have been stopped to all but the IMS system that needs affinity, start the link from the remote IMS system.
- In each IMS system in the link connection is to be prevented, stop logons to the generic resource group on the physical link.
  a) You can stop logons in an IMS system by issuing either one of the following IMS commands for each IMS system with which the link connection needs to be prevented:
     – The IMS type-2 command **UPDATE MSPLINK** NAME(*msplink_name*) STOP(GENLOGON)
     – The IMS type-1 command **/PSTOP MSPLINK** *msplink_name*
  b) After logons to the generic resource group have been stopped for all IMS systems in the group except the IMS system that the link will connect to, start the link from the remote IMS system.

Related reference

UPDATE IMSCON TYPE(MSC) command (Commands)
UPDATE MSLINK command (Commands)
/RSTART command (Commands)
STOPMSC command (Commands)
STARTMSC command (Commands)
IMS Connect UPDATE MSC command (Commands)

Persistence of MSC link affinity in a TCP/IP generic resource group

The affinity of an MSC link to a specific IMS system in a TCP/IP generic resource group is established when the link is started and is cleared when the link is stopped normally and assumes an NRE or COLD status.

In the event of a failure of an IMS system that has affinity, IMS Connect usually receives notification of the failure from the Structured Call Interface (SCI). After it is notified of the failure, IMS Connect terminates the link path to the failed IMS system and clears the link affinity.

If IMS Connect does not receive notification of the failure, the link affinity can persist. Link affinities that persist must be cleared manually in IMS Connect by issuing an IMS Connect command before the physical link can be restarted on another IMS system.

Related tasks

"Managing MSC links in a TCP/IP generic resource group" on page 726
When MSC is used with TCP/IP generic resources, each link that connects to the TCP/IP generic resource group has affinity to a specific IMS system in the group.

"Clearing MSC link affinity in a TCP/IP generic resource group" on page 728
MSC logical link parallel sessions have affinity to specific IMS systems in TCP/IP generic resource group. While affinity persists, you cannot reassign links to another IMS system. To clear the affinity of a link in a TCP/IP generic resource group, you must stop all logical link parallel sessions on the physical link.

Related reference

UPDATE IMSCON TYPE(MSC) command (Commands)

STOPMSC command (Commands)

STARTMSC command (Commands)

IMS Connect UPDATE MSC command (Commands)

Clearing MSC link affinity in a TCP/IP generic resource group

MSC logical link parallel sessions have affinity to specific IMS systems in TCP/IP generic resource group. While affinity persists, you cannot reassign links to another IMS system. To clear the affinity of a link in a TCP/IP generic resource group, you must stop all logical link parallel sessions on the physical link.

To clear the affinity of an MSC logical link to an IMS system in a TCP/IP generic resource group:

- Stop the logical link by issuing any one of the following IMS commands or command sequences:
  - The IMS type-2 format command **UPDATE MSPLINK** NAME(*mslink_name*) STOP(COMM)
  - The IMS type-2 format command **UPDATE MSPLINK** NAME(*mslink_name*) STOP(COMM) OPTION(FORCE) SET(SYNCOPT(COLDSESS))
  - The IMS type-1 format command **/PSTOP LINK** *link_num*
  - The command sequence of the IMS type-1 format command **/PSTOP LINK** *link_num* and then the type-1 command **/CHANGE LINK** *link_num* with either the SYNCSESS COLDSESS or FORCSESS COLDSESS keywords.

After the link is stopped in IMS, the link affinity is cleared.

Related tasks

"Managing MSC links in a TCP/IP generic resource group" on page 726
When MSC is used with TCP/IP generic resources, each link that connects to the TCP/IP generic resource group has affinity to a specific IMS system in the group.

Related reference

QUERY IMSCON TYPE(MSC) command (Commands)

/RSTART command (Commands)

UPDATE IMSCON TYPE(MSC) command (Commands)

VIEWMSC command (Commands)

STOPMSC command (Commands)

STARTMSC command (Commands)
IMS Connect QUERY MSC command (Commands)
IMS Connect UPDATE MSC command (Commands)

Clearing MSC link affinity in IMS Connect

Normally, IMS Connect automatically clears MSC TCP/IP link affinity when all of the MSC logical links on a physical link are terminated normally in IMS. If the affinity persists in IMS Connect, you can clear the affinity by stopping and restarting the physical link path in IMS Connect.

To clear a persistent affinity of an MSC physical link to an IMS system in IMS Connect:

- If for some reason the link affinity does not clear in IMS Connect, you can stop the physical link path in IMS Connect by issuing any one of the following IMS Connect commands:
  - The IMS type-2 format command **UPDATE IMSCON** TYPE(MSC) NAME*(lclPlkid*) STOP(COMM)
  - The WTOR format command **STOPMSC** *lclPlkid*
  - The z/OS MODIFY format command **UPDATE MSC** NAME(*lclPlkid*) STOP(COMM)

Before the link can be used again, you must restart the link in IMS Connect by issuing one of the following commands:

- The IMS type-2 format command **UPDATE IMSCON** TYPE(MSC) NAME*(lclPlkid*) START(COMM)
- The WTOR format command **STARTMSC** *lclPlkid*
- The z/OS MODIFY format command **UPDATE MSC** NAME(*lclPlkid*) START(COMM)

Related tasks

"Managing MSC links in a TCP/IP generic resource group" on page 726
When MSC is used with TCP/IP generic resources, each link that connects to the TCP/IP generic resource group has affinity to a specific IMS system in the group.

Related reference

QUERY MSLINK command (Commands)
/PSTOP command (Commands)

XRF, MSC, and TCP/IP generic resources

When an XRF pair of active and alternate IMS systems use TCP/IP generic resources to support MSC, the MSC physical link and all of its parallel sessions have affinity to the active IMS system.

To support TCP/IP generic resources, the same GENIMSID parameter must be specified in the DFSDCxxx PROCLIB member of both the active and alternate IMS systems in the XRF pair.

When a takeover occurs, IMS switches the affinity of the logical links from the active IMS system to the alternate IMS system, which becomes the new active IMS system.

IMS also notifies IMS connect of the failure of the active IMS system within the IMSplex. IMS Connect terminates the sockets that were active at the time of the failure and releases the affinity between the MSC physical link and the failed IMS system.

To verify that the affinity to the failed IMS system has been released, issue an IMS Connect command to display the status of the physical link. If IMS Connect has not released the affinity status, such as might happen if IMS Connect was not notified of the takeover, clean up the sockets by stopping and restarting the MSC link by issuing the appropriate IMS Connect commands.

After the affinity status is deleted in IMS Connect, you can start the MSC links on the new active IMS system. After the link is started in the new active IMS system, the link will have affinity with the new active IMS system and IMS Connect will show the new affinity status.

An XRF alternate IMS system will not respond to a restart request from a remote IMS system outside of the IMSplex. MSC links cannot be started on an alternate IMS system until the takeover completes and the alternate IMS system becomes the new active IMS system.

Related tasks

Defining a TCP/IP generic resource group for MSC (System Definition)

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

QUERY IMSCON TYPE(MSC) command (Commands)

UPDATE IMSCON TYPE(MSC) command (Commands)

VIEWMSC command (Commands)

STOPMSC command (Commands)

STARTMSC command (Commands)

IMS Connect QUERY MSC command (Commands)

IMS Connect UPDATE MSC command (Commands)

VTAM Generic Resources (VGR) and MSC

MSC IMS systems outside of an IMSplex can use a VTAM generic resource group name (GRSNAME) to establish MSC link sessions with the IMSplex instead of using the APPLID of a specific IMS system in the IMSplex.

When you use a VTAM GRSNAME, you can move MSC link sessions between IMS systems within the IMSplex without having to change the VTAM node name on the NAME parameter of the CREATE MSPLINK command or in the MSPLINK stage 1 system definition macro for each IMS system outside the IMSplex.

Using a VTAM GRSNAME can also make cloning IMS systems within an IMSplex easier because the same MSC link definition can be used by each IMS system within the IMSplex to connect to a remote IMS system. Each parallel link in the IMSplex can use a common GRSNAME to establish a session with an IMS system outside of the IMSplex. Other link attributes, such as partner ID, SYSIDs, and so forth, can also be cloned across the IMS systems within an IMSplex.

The use of VTAM generic resources in an IMSplex is transparent to the external MSC system. The external MSC system need only specify the GRSNAME that is used by the IMSplex, instead of a specific IMS system APPLID name, on the NAME parameter of the CREATE MSPLINK command or the MSPLINK macro. In the IMSplex, the MSC VGR support is enabled by specifying MSCVGR=Y (MSCVGR=N is the default) in the DFSDCxxx PROCLIB member of each IMS system in the IMSplex. The IMSplex must also have VTAM generic resources enabled.

When MSC VGR support is enabled, also specify GRMESTAE=Y (the default) in the DFSDCxxx PROCLIB member to ensure that IMS deletes VTAM affinity if an IMS abend should occur. The type-1 command **/DISPLAY AFFIN LINK** and the type-2 command **QUERY MSLINK SHOW(AFFIN)** display current affinities between MSC links and the IMS systems in an IMSplex.

Related concepts

Planning for VTAM generic resource groups (System Administration)

Related reference

DFSDCxxx member of the IMS PROCLIB data set (System Definition)

CREATE MSLINK command (Commands)

QUERY MSLINK command (Commands)

/DISPLAY AFFIN command (Commands)

TM and MSC Message Routing and Control user exit routine overview

Message routing is automatic, according to the defined scheme, unless you use the TM and MSC Message Routing and Control user exit routine (DFSMSCE0), which provides routing options and control of messages.

The DFSMSCE0 exit routine provides a single parameter list and the ability to append optional user prefixes to messages for customizing message routing and security.

The DFSMSCE0 exit routine also provides an IMSplex affinity routing option for IMSplexes with shared queues. Affinity routing establishes an affinity between a transaction instance and a destination IMS system specified by the exit routine. When the DFSMSCE0 exit routine assigns an affinity to a transaction and the transaction is inserted to the shared queue, only the destination IMS system is notified.

Note: Using this exit routine does not require Multiple Systems Coupling (MSC), though most of its options work only when MSC is enabled.

This routing exit routine is called before the message destination is final. The entry points of the exit routine are:

- **Terminal routing (TR):** Receives control when a message is received from a terminal.
  - VTAM messages (TRVTAM)
  - APPC messages (TRAPPC)
  - OTMA messages (TROTMA)
- **Link receive (LR):** Receives control when a message is received on a MSC link.
  - Local transaction messages (LRTRAN)
  - Local LTERM messages (LRLTERM)
  - Local direct routing messages (LRDIR)
  - Intermediate messages (LRINT)
- **Program routing (PR):** Receives control when the application program issues a CHNG or ISRT call to insert a message.
  - Application program CHNG call (PRCHNG)
  - Application program INST call (PRINST)

The TM and MSC Message Routing and Control User Exit routine is loaded during IMS initialization, provided it exists in the JOBLIB, STEPLIB, or LINKLIST library that is concatenated in front of IMS.SDFSRESL. No system definition or startup parameter modules are needed to invoke this exit routine.

You can append optional user-defined prefixes to messages. Message prefixes can be used, for example, to customize message security, user accounting, and statistics requirements, and to increase routing control by allowing communications among exit routines. This user message prefix segment can be added to the message and updated as the message is routed through the MSC/TM network as each exit routine entry point is called. These other exit routines can read or update the prefix segment. The user prefix segment can be used offline. The user prefix size is limited to 512 bytes, and the total message prefix size is limited to the large message queue LRECL.

The TM and MSC Message Routing and Control User Exit routine uses a common parameter list interface (DFSMSCEP) for all of the entry points. DSECTS are provided to reference all parameter fields. You can select the entry points at which the exit routine should take control by changing and reassembling pointers in the DFSMCCSV macro in the front of the user exit module. IMS calls the exit routine if the entry point exists.

Recommendation: Provide a DSECT DFSMSCEP macro for common reference to the parameter fields to make it easier in the future to pass to the exit routines additional data, such as performance or path availability data for use in making routing decisions. Data might include SIO rates, response times, or queue counts.

Terminal routing

The terminal routing entry points of the TM and MSC Message Routing and Control User Exit routine exist in the input system. The exit routine is called when a message is received from a terminal. The exit routine can inspect the specified destination (LTERM or transaction code) and, if specified, reject it or change it to any local or remote destination. If the exit routine does not change the destination, the originally specified destination is used for routing. The exit routine can also override a local LTERM or transaction to route a message to a remote IMS instead.

IMS does not call the exit routine for commands or from a terminal that is continuing a conversation.

In a configuration using horizontal partitioning, the exit routine can be used to evaluate all input messages and route them to the appropriate processing system based on information in the first segment of the input message. If transactions and links are appropriately defined, the exit routine can also be used to set a common destination for a set of input messages. On arriving at the destination system, the messages are processed according to their individual transaction codes.

Link receive

The Link Receive entry points of the TM and MSC Message Routing and Control User exit routine (DFSMSCE0) can change the transaction code or LTERM name of a message when IMS receives the message from an MSC link. DFSMSCE0 can request that the message be processed locally in the current IMS system or it can reroute the message to a different remote IMS system. DFSMSCE0 can inspect the transaction code or the LTERM name that is used as the destination and, if specified, reject it or change it to another destination with the same attributes. The exit routine can also examine the first segment of the message to determine what the new transaction code should be. If the exit routine does not change the transaction code, the destination remains the location that is specified by the original transaction code. DFSMSCE0 can also determine what security to use, if any, on a message by message basis.

Program routing

By using the program routing points of the TM and MSC Message Routing and Control User exit routine (DFSMSCE0), you can control the routing of a message when an application program issues a CHNG or ISRT call. With these entry points, you can change the destination of the message or request to reject a message. The DFSMSCE0 exit routine can also request that the message be processed locally in the current IMS system or it can reroute the message to a different remote IMS system.

You can use the exit routine to avoid defining unique names for remote LTERMs and transactions. With the exit routine, you can have the same name for terminals throughout your MSC network. You can use this exit routine to execute the DL/I CHNG or ISRT call for the application program, and to change the destination from local to remote.

Related reference

TM and MSC Message Routing and Control User exit routine (DFSMSCE0) (Exit Routines)

MSGQUEUE macro (System Definition)

The IMSplex affinity routing option of the DFSMSCE0 exit routine

In IMSplexes with shared queues, the IMSplex affinity routing option of the TM and MSC Message Routing and Control user exit routine (DFSMSCE0) can establish an affinity between a transaction instance and an IMS system in which the transaction is started with the affinity option.

MSC must be enabled in the IMS systems that use the IMSplex affinity routing option of the DFSMSCE0 exit routine.

When the DFSMSCE0 exit routine assigns an affinity to a transaction and the transaction is inserted to the shared queues, only the IMS system specified by the DFSMSCE0 exit routine is notified.

The IMSplex affinity routing option of the DFSMSCE0 exit routine enables back-end IMS systems in an IMSplex to process both APPC synchronous transactions and OTMA send then commit (CM1) transactions. The IMSplex affinity routing option overrides restrictions that would otherwise prevent the APPC and OTMA CM1 transactions from processing on back end IMS systems in the IMSplex. When the affinity option is invoked, APPC and OTMA transactions run in a mode that disconnects the transaction message from the client, similar to the mode in which MSC routes messages to remote IMS systems in an MSC network.

The DFSMSCE0 exit routine can be called to establish affinity at the following exit points:

Terminal routing (TR)
The DFSMSCE0 exit routine receives control and can set affinity for a message that is received from a terminal by an IMS system in an IMSplex.

Link receive (LR)
The DFSMSCE0 exit routine receives control and can set affinity for a message that is received from an MSC link by an IMS system in an IMSplex.

Program routing (PR)
The DFSMSCE0 exit routine receives control and can set affinity for a message when an application program running in an IMS system in an IMSplex issues a CHNG or ISRT call to insert the message.

The DFSMSCE0 exit routine establishes affinity for a transaction instance by appending the IMS ID or, in an XRF complex, the RSENAME of the destination IMS system to the shared queue name (SQNAME) in the message prefix.

Before a transaction instance can be processed by the IMS system with which it has affinity, the transaction type must be registered as having affinity on the IMS system.

To start a transaction with the affinity option in an IMS system, specify either of the following commands on the IMS system that processes the transaction:

- The type-2 command **UPDATE TRAN** NAME(*tranname*) START(SCHD) OPTION(AFFIN)
- The type-1 command **/START TRAN** *trancode* AFFINITY

IMS maintains the affinity status of a transaction type in an IMS system across warm starts and emergency restarts, but not across either a cold start or an export and import of a transaction definition to a resource definition data set.

To display transactions on the shared queues that have affinity to an IMS system, you can issue either of the following commands with the appropriate keywords:

- The type-2 command QUERY TRAN QCNT(GT,0) SHOW(AFFIN)
- The type-1 command /DIS TRAN ALL QCNT

After an IMS system retrieves a transaction with affinity from the shared queue, the affinity status of the transaction is displayed under LclStat as AFFIN. To see which transactions have local affinity status on an IMS system, you can issue the following type-2 command:

- QUERY TRAN NAME(*tranname* | *) SHOW(STATUS)

For example, if you issue `QUERY TRAN NAME(APOL11 APOL12) SHOW(STATUS)`, the command output might look like the following:

```
Trancode MbrName   CC  LclStat
APOL11   IMS1      0
APOL12   IMS1      0   AFFIN
```

Related reference
TM and MSC Message Routing and Control User exit routine (DFSMSCE0) (Exit Routines)

Using the IMSRSC repository with MSC

For IMS systems that are capable of Multiple Systems Coupling (MSC), the IMSRSC repository can be defined among the IMS systems.

IMSRSC repository definitions and MSC

If the MSC links are within an IMSplex, all IMS systems share the same repository defined for the IMSplex. If the MSC links are between IMSplexes, and if the IMSplexes are within the same z/OS sysplex, each IMSplex can either have its own repository or share the same repository. The repositories in these different IMSplexes can be managed by the same active Repository Server (RS) address space or by a different active RS address space.

The following MSC resources can be maintained in the repository:

- Remote transactions and transaction descriptors
- Physical and logical links
- Link paths
- Remote logical terminals (LTERMs)

If a remote transaction or a transaction descriptor has a program name associated with it, the program resource is not required to exist in the repository. The program resource associated with the transaction or the transaction descriptor that is defined as local must exist in the repository.

Recommendation: When MSC links are between IMSplexes within the same z/OS sysplex, define one RS and a separate repository for each IMSplex because the separate repositories are easier to manage.

If the MSC links are between IMSplexes, and if the IMSplexes are in different z/OS sysplexes, each IMSplex must have its own repository and RS address space. The repositories of these different IMSplexes must be managed by different RS address spaces.

For the resource definitions that have different attributes among IMS systems (such as the SIDR and SIDL values for each MSC-capable system), the stored resource definitions in the repository consist of a generic section (for the common attributes) and an IMS-specific section (for the attributes that are different for each IMS system).

Related concepts

Overview of the IMSRSC repository (System Definition)

How SIDR and SIDL values for remote trans and descriptors are stored

For the resource definitions that have different attributes among IMS systems (such as the SIDR and SIDL values for each MSC-capable system), the stored resource definitions in the repository consist of a generic section (for the common attributes) and an IMS-specific section (for the attributes that are different for each IMS system).

For remote transactions and transaction descriptors, the SIDR and SIDL values are not the same. The SIDR and SIDL values for each IMS system are maintained in the repository in an IMS-specific section. Each IMS has its own specific section for the SIDR and SIDL values. For local transactions and transaction descriptors, the SIDR and SIDL values are set to 0 in the repository in the generic section. When the stored resource definition is imported from the repository either during AUTOIMPORT processing or during processing of the **IMPORT** command, the SIDR and SIDL values are set to the lowest local SID value of the IMS system where the runtime resource definition is created.

The SIDR and SIDL values in the repository for remote transactions and transaction descriptors remain in the IMS-specific section and are not updated or collapsed into the generic section when the **EXPORT** command with the SET(IMSID(*)) keyword is issued. To modify the SIDR and SIDL values for remote transactions and transaction descriptors, you must issue the **EXPORT** command with a specific IMS ID specified on the SET(IMSID) keyword or the default SET(IMSID()) keyword to route the command to the IMS whose definitions are to be exported.

Related concepts

Overview of the IMSRSC repository (System Definition)

Maintaining MSC resources in the IMSRSC repository

Maintain MSC resource definitions in the IMSRSC repository to store the definitions in a single, centralized location for all members of an IMSplex. Maintaining MSC resource definitions in the IMSRSC repository also enables the definitions to be saved across an IMS cold start.

If you use the IMSRSC repository to store dynamically defined MSC resources, ensure that automation and operational procedures that issue commands for MSC resources use type-2 commands, which specify link names, instead of type-1 commands, which specify link numbers. For example, instead of using the **/RSTART LINK 10** command to start a link, use the `UPDATE MSLINK NAME(`*`logicallinkname`*`) START(COMM)` command. During stage-1 system generation, the IMS

system assigns numbers to logical links in the order in which the links are generated. However, the numbers for links are not stored in the IMSRSC repository. If logical links are referenced by using link numbers and are automatically imported from the IMSRSC repository, the numbers of the links are likely to change at the next IMS cold start.

Before you create and maintain MSC resources in the IMSRSC repository, ensure that all of the following prerequisites are met:

- The IMSRSC repository is defined.
- Dynamic resource definition for MSC resources is enabled.
- The IMSRSC repository is enabled for MSC resources.
- MODBLKS=DYN is specified in either or both of the following locations:
  - The COMMON_SERVICE_LAYER section of the DFSDFxxx PROCLIB member
  - The DFSCGxxx member of the IMS PROCLIB data set
- AUTOEXPORT=AUTO or AUTOEXPORT=REPO is specified in the DYNAMIC_RESOURCES section of the DFSDFxxx PROCLIB member.
- AUTOIMPORT=AUTO or AUTOIMPORT=REPO is specified in the DYNAMIC_RESOURCES section of the DFSDFxxx PROCLIB member.

Creating MSC resources in the IMSRSC repository

To create MSC resources in the IMSRSC repository, issue the **CREATE** command for the type of MSC resource that you want to create. You must create MSC resources in the following order for the MSC link to be usable:

1. CREATE MSPLINK
2. CREATE MSLINK
3. CREATE MSNAME
4. CREATE LTERM

Issue the **CREATE** command to each IMS in each IMSplex for which the resource definitions are to be exported to the repository. You can use the ROUTE keyword on the OM API to specify the IMS to which the command is routed.

The MSC resource definitions that you created are automatically exported to the IMSRSC repository at the end of the next IMS checkpoint or before the IMS shutdown checkpoint.

At the end of the automatic export processing to the IMSRSC repository, the X'22' map byte X'51' automatic export complete log record is written.

Updating MSC resources in the IMSRSC repository

To update MSC resources in the IMSRSC repository, use the appropriate **UPDATE** command for the type of resource definition that you are updating.

The UPDATE command must be issued to each IMS in each IMSplex for which the resource definitions are to be exported to the repository. You can use the ROUTE keyword on the OM API to specify the IMS to which the command is routed.

Automatic export processing is done at the end of each IMS normal checkpoint and before the IMS shutdown checkpoint. IMS automatically exports any changes in resource definitions for MSC resources since the last automatic export.

Deleting MSC resources from the IMSRSC repository

To delete resource and descriptor definitions from the IMSRSC repository, issue the **DELETE DEFN** command. Issuing the **DELETE DEFN** command ensures that the definitions remain deleted in IMS™ across a cold start, and that they are not imported from the repository during cold start processing

Delete MSC resource definitions from the IMSRSC repository in the following order:

1. Remote transaction definitions
2. Remote logical terminal (LTERM) definitions
3. MSC logical link paths (MSNAMEs)
4. MSC logical links (MSLINKs)
5. MSC physical links (MSPLINKs)

Important: When you delete an MSC resource of one type and re-create the resource as another type, first delete the resource from the local IMS system, and then delete the resource from the IMSRSC repository. If the MSC resource of the original type is not deleted first locally and then from the IMSRSC repository, the IMS system might fail to re-create the original resource as another resource type.

If you delete runtime resource definitions by using a **DELETE** command and then cold start IMS, the deleted resource definitions reappear if the resource definitions are automatically imported from the original DFSCLL3x member of the IMS.SDFSRESL data set instead of the IMSRSC repository that contains the most current definitions.

Exporting MSC resources to the IMSRSC repository

At the end of the next IMS normal checkpoint or before the IMS shutdown checkpoint, IMS automatically exports to the IMSRSC repository the newly created or updated MSC resource definitions since the last automatic export. The IMS checkpoint can be initiated either by issuing the **/CHECKPOINT** command or automatically by the IMS system.

If you use the **/CHECKPOINT** command to initiate an IMS checkpoint, the command must be routed to each IMS in each IMSplex in which the MSC resources are defined.

Importing MSC resources from the IMSRSC repository

If MSC resource definitions exist in the IMSRSC repository, the definitions are imported from the repository at the next IMS cold start.

If you are using channel-to-channel (CTC) links, consider removing the DD definitions for the CTC links from the IMS JCL before you import the MSC resources from the IMSRSC repository. This allows the CTC addresses that are defined to be used for the CTC links that are imported from the IMSRSC repository.

If an IMS system is cold started on a different z/OS system than the z/OS system where a channel-to-channel link was created and exported, and that channel-to-channel address is not defined on the new z/OS system, the CTC link open fails and omits issuing the DFS2168I `CONNECTION ESTABLISHED ON LINK x` message. However, the automatic import continues. You can use the `UPDATE MSPLINK SET(ADDR(addr))` command to change the address of the MSC physical to an address that is valid for that z/OS system.

Removing the DFSCLL3x and DFSCLR0x members and updating the DFSCLC0x member of the IMS.SDFSRESL data set

If you use the IMSRSC repository to store dynamically defined MSC resources, the DFSCLL3x and DFSCLR0x members of the IMS.SDFSRESL data set are no longer required. Because the DFSCLC0x member might contain non-MSC resources, the DFSCLC0x might still be required. After you are satisfied with the setup of your DRD environment for MSC resources, the DRD environment is running successfully, and your MSC resources are exported to the IMSRSC repository, you can remove the DFSCLL3x and DFSCLR0x members. For the DFSCLC0x member, update the member to remove MSC logical link path definitions. If automatic import is enabled, MSC resource definitions are imported during IMS cold start from the repository that contains the most current data. However, you can continue to use the DFSCLL3x, DFSCLR0x, and DFSCLC0x members as the source for your MSC resource definitions, instead of a repository. If you continue to use the DFSCLL3x, DFSCLR0x, and DFSCLC0x members, keep your system definition macros synchronized with the changes you make dynamically using DRD commands.

Synchronizing the DFSCLL3x member of the IMS.SDFSRESL data set

If you migrate to using the IMSRSC repository for dynamically defined MSC resources but continue to use the DFSCLL3x, DFSCLR0x, and DFSCLC0x members of the IMS.SDFSRESL data set, keep resource definitions that are in the members synchronized with the resource definitions that are in the repository. This synchronization enables you to maintain viable DFSCLL3x, DFSCLR0x, and DFSCLC0x members if you must disable DRD and fall back to using the system generation process for MSC resources. To keep your DFSCLL3x, DFSCLR0x, and DFSCLC0x members synchronized with your online definitions, update your static macro definitions with the changes that you make dynamically using type-2 commands. When changes are made dynamically, perform a MSC system definition to add, change, or delete resources from the DFSCLL3x, DFSCLR0x, and DFSCLC0x members.

Related concepts

Deleting runtime resource and descriptor definitions (System Definition)

Related tasks

Enabling the IMSRSC repository for MSC resources (Communications and Connections)

Defining the IMSRSC repository (System Definition)

Creating MSC resource definitions in the IMSRSC repository (System Definition)

Enabling dynamic definition for MSC resources (System Definition)

Exporting MSC resource definitions to an IMSRSC repository (System Definition)

Updating MSC resource definitions in the IMSRSC repository (System Definition)

Related reference

CREATE LTERM command (Commands)

CREATE MSLINK command (Commands)

CREATE MSNAME command (Commands)

CREATE MSPLINK command (Commands)

/CHECKPOINT command (Commands)

DELETE DEFN command (Commands)

UPDATE MSLINK command (Commands)

UPDATE MSNAME command (Commands)

UPDATE MSPLINK command (Commands)

MSC section of the DFSDFxxx member (System Definition)

DYNAMIC_RESOURCES section of the DFSDFxxx member (System Definition)

Creating or updating MSC resources in the repository

You can create or update MSC resources in the IMSRSC repository for the IMSplex.

Before you create or update MSC resource definitions in the IMSRSC repository, ensure that the following prerequisites are met:

- The IMSRSC repository is defined.
- Dynamic resource definition for MSC resources is enabled.
- The IMSRSC repository is enabled for MSC resources.
- AUTOEXPORT=AUTO or REPO is specified in the DYNAMIC_RESOURCES section of the DFSDFxxx member of the IMS PROCLIB member.
- MODBLKS=DYN is specified either in the COMMON_SERVICE_LAYER section of the DFSDFxxx PROCLIB member or in the DFSCGxxx member of the IMS PROCLIB data set.

To create or update MSC resources in the repository for the IMSplex:

1. Create or update MSC resources locally at the IMS system by using the appropriate CREATE or UPDATE command.

   At the next IMS checkpoint, the MSC resource definitions that you created or updated since the last automatic export are exported to the IMSRSC repository. The IMS checkpoint can be initiated either by issuing the /CHECKPOINT command or automatically by the IMS system.

2. For remote transactions and transaction descriptors, you can issue the **EXPORT DEFN TARGET(REPO)** command, which defaults to the SET(IMSID()) keyword, to route the command to the IMS system whose resource definitions are to be exported.
3. Repeat these steps at each IMS system where the MSC resource definitions are, or will be, defined. The repository will maintain the attributes of the MSC resources for each IMS.

Related reference

EXPORT command (Commands)

DYNAMIC_RESOURCES section of the DFSDFxxx member (System Definition)

Updating transactions from remote to local by using the repository

You can update a transaction or transaction descriptor from remote to local in an IMSplex by using the IMSRSC repository.

To update transactions and transaction descriptors from remote to local in an IMSplex by using the repository:

1. Issue the **UPDATE TRAN** command or the **UPDATE TRANDESC** command with the REMOTE, SIDR, and SIDL keywords. In the command, specify REMOTE(N), and set the SIDR and SIDL values to the same local system identifier (SYSID).
2. Issue the **EXPORT DEFN TARGET(REPO)** command, which defaults to the SET(IMSID()) keyword, to route the command to the IMS system whose resource definitions are to be exported. Or, if automatic export to the IMSRSC repository is enabled with AUTOEXPORT=AUTO or REPO, ensure that AUTOEXPORT_IMSID is set to THIS_IMS.
3. Repeat these steps at each IMS system where the transaction or the transaction descriptor is to be defined as local. The repository will remove the remote attributes for each IMS in the repository.

Related reference

EXPORT command (Commands)

UPDATE TRAN command (Commands)

UPDATE TRANDESC command (Commands)

DYNAMIC_RESOURCES section of the DFSDFxxx member (System Definition)

Updating transactions from local to remote by using the repository

You can update transactions and transaction descriptors from local to remote in an IMSplex by using the IMSRSC repository.

To update transactions and transaction descriptors from local to remote in an IMSplex by using the repository:

1. Issue the **UPDATE TRAN** command or the **UPDATE TRANDESC** command with the REMOTE keyword and either the SIDR and SIDL keywords or the MSNAME keyword. In the command, specify REMOTE(Y) and either set the SIDR value to a remote SYSID and the SIDL value to a local SYSID, or specify the name of a logical link path by using the MSNAME keyword.
2. Issue the **EXPORT DEFN TARGET(REPO)** command, which defaults to the SET(IMSID()) keyword, to route the command to the IMS system whose resource definitions are to be exported. Or, if automatic export to the IMSRSC repository is enabled with **AUTOEXPORT=***AUTO* or *REPO*, ensure that **AUTOEXPORT_IMSID** is set to *THIS_IMS*. Both parameters (**AUTOEXPORT** and **AUTOEXPORT_IMSID**) can be found in the DYNAMIC_RESOURCES section of the DFSDFxxx member.
3. Repeat these steps at each IMS system where the transaction or the transaction descriptor is to be defined as remote. The repository will maintain the remote attributes for each IMS.

Related reference

EXPORT command (Commands)

UPDATE TRAN command (Commands)

UPDATE TRANDESC command (Commands)

DYNAMIC_RESOURCES section of the DFSDFxxx member (System Definition)

Part 9. ODBA and DRA connections

The Open Database Access (ODBA) and database resource adapter (DRA) interfaces provide direct access to IMS databases from applications running on the same z/OS image is IMS.

Application programs that use DRA or ODBA and DRA include:

- CICS Transaction Server for z/OS
- ODBA z/OS applications, such as Db2 for z/OS stored procedures

© Copyright IBM Corp. 1974, 2020

Chapter 40. Accessing IMS databases with CICS

CICS application programs can access DL/I databases through DBCTL. This means that DBCTL satisfies CICS DL/I requests by means of the CICS-DBCTL interface.

CICS/ESA and CICS Transaction Server for z/OS provide support for IMS DBCTL.

Installing data sharing with CICS

CICS users can optionally use the IMS data sharing facility. The Internal Resource Lock Manager (IRLM) is mandatory for block-level data sharing, but not for database-level sharing.

Installing CICS for use with IMS intersystem communication

Information about defining CICS as an Intersystem Communication node and defining CICS tables to be compatible with IMS in an Intersystem Communication network is provided in Part 7, “Intersystem Communication (ISC),” on page 443.

Related concepts

Data sharing in IMS environments (System Administration)

Coding considerations for PSBs

This topic provides some guidelines for coding PSBs for the DRA when using CICS.

PSBs for Online Transactions

- A PSB is needed for each online program that accesses DL/I databases.
- The name of the PSB specified in the PSBNAME= keyword of the PSBGEN macro must be exactly the same as one of the entries in the PDIR.
- The name of the PSB must also be the same as the name specified in the scheduling call issued by an online transaction. If the online transaction does not specify any name in the scheduling call, the name of the PSB must be exactly the same as the name of the program associated with the transaction in the CICS program control table (PCT). For example, assume transaction X is associated with program Y. If program Y links or transfers control to program Z, which issues the scheduling call without specifying any PSB name, the default name for the PSB is Y, because it is program Y that is associated with transaction X in the PCT.
- The CMPAT=YES option on the PSBGEN statement can be omitted.
- There is no specific maximum size for an individual PSB that can be used in CICS/ESA. However, a limitation is set by the PSB pool size in DBCTL.

Additional Processing Intent Options (PROCOPT)

Two additional types of processing intent can be specified with the PROCOPT= keyword of the PCB or SENSEG statement. The two additional options are O and E.

PROCOPT=O specifies "read without integrity"; no dynamic enqueue is done by resource lock management for calls against the database. With the G intent option, you can specify GO, GOP, GON, or GOT. This option is only valid for the PCB statement. Read note number 1 under Attention for more information.

PROCOPT=E forces exclusive use of this database or segment by the online transaction. Other application programs scheduling a PSB referring to this database/segment wait during their scheduling process. No dynamic enqueue by resource lock management is done, but dynamic logging of database updates will be done. PROCOPT E can be specified with G, I, D, R, and A.

Attention:

© Copyright IBM Corp. 1974, 2020

1. The PROCOPT=O option affects integrity in reading and using uncommitted data. When you specify this option, IMS does not check the ownership of the segments returned. This means that the read-only user might have access to a segment that had been updated by another user. If the updating user then abends and is backed out, the read-only user would have seen a segment that no longer exists in the same form in the database. Consequently, if you specify this option, do not use the data that is read as a basis for updating records in any database.
2. An abend might occur with PROCOPT=GO if another program updates pointers when this program is following the pointers. Pointers are updated during insert, delete, and backout operations.
3. If PROCOPT=O in the PCB statement, the SENSEG statement must not specify a PROCOPT of I, R, D, H, or A.
4. If the O or E option is used, it must be coded immediately after the associated function code; for example GO, not OG.

Related reference

Program Specification Block (PSB) Generation utility (System Utilities)

Using sequential buffering

You can use sequential buffering with CICS.

To use sequential buffering with CICS, you need to do two things:

- Put an SBONLINE control statement in the //DFSVSMxx file. SBONLINE allows sequential buffering to be used.
- Specify programs that are to use sequential buffering. You can do this by coding during PSBGEN an SB= keyword on the PCB macro.

Related reference

Full-function or Fast Path database PCB statement (System Utilities)

Specifying sequential buffering for an online system (System Definition)

CICS connected to DL/I

CICS can provide DL/I database support by using IMS.

There are two ways in which you can use DL/I support with CICS:

- Through CICS remote DL/I support, also known as function shipping
- Through DBCTL

For CICS/ESA, remote DL/I support and DBCTL support are included in the pregenerated CICS.

Configuring CICS CCTL connections to IMS DBCTL systems

To connect CICS to databases managed by IMS DB, you must define the CICS resources and initialize the IMS DBCTL system.

- Define CICS resources

  If you use DBCTL exclusively, define the PSBs and DMBs in IMS by using one of the following methods:

  – Use the APPLCTN and DATABASE macros.
  – In systems that have dynamic resource definition (DRD) enabled, use the commands `CREATE DB` and `CREATE PGM`.
  – In systems that have dynamic resource definition (DRD) enabled, use the Program Creation user exit routine (PGMCREAT) to define the PSBs that are associated to programs scheduled in BMP and JBP dependent regions.

If you want to function-ship requests to a remote CICS system, in which the database manager can be DBCTL or remote DL/I (function-shipping), you need to generate a PDIR only.

CICS routes DL/I requests to remote DL/I or DBCTL according to the PSB that is named. If the PSB does not appear in the CICS PDIR, and CICS is connected to DBCTL, CICS routes the request to DBCTL.

Related reading: Refer to *CICS Transaction Server for z/OS CICS System Definition Guide* and *CICS Transaction Server for z/OS CICS Resource Definition Guide* for information on defining PDIRs and DDIRs.

- Initialize IMS DBCTL

  Use the procedure library member DBC to initialize the DBCTL subsystem. Also generated are procedures for DBRC and DL/I, which are used to initialize the DBRC and DL/I address spaces. The DBRC and DL/I procedures are started automatically by DBCTL during DBCTL address space initialization. All three procedures use positional parameters on the EXEC statement:

  ```
  PARM='region type, parm1, parm2, parm3,...'
  ```

  The region types specified are:

  ```
  PARM='DBC' for DBCTL
  PARM='DRC' for DBRC
  PARM='DLS' for DLISAS
  ```

  When all three address spaces have been initialized successfully, DBCTL issues:

  ```
  DFS989I DBCTL READY
  ```

  When the "READY" message is received, the IMS console operator enters a `/START` command. The commands options are:

  – `/NRESTART CHECKPOINT 0` for a cold start with no previous shutdown
  – `/NRESTART` for a warm start
  – `/ERESTART` for an emergency restart after a failure

  Related reading: See *CICS Transaction Server for z/OS IMS Database Control Guide* for additional details.

CICS tasks

The database resource adapter (DRA) is the interface between DBCTL and the transaction management subsystem. You need to put the DRA startup parameter table (DFSPZPxx) and the DRA startup router program (DFSPRRC0) in the CICS STEPLIB data set.

This topic describes CICS tasks associated with using DBCTL and CICS. The high level tasks include:

1. Define DRA resources

   Example:

   ```
   //STEPLIB  DD DSN=CICSTSxx.CICS.SDFHAUTH,DISP=SHR
   //         DD DSN=IMS.SDFSRESL,DISP=SHR
   ```

   Related reading: For more information on the DRA startup table see *IMS Version 15 Application Programming APIs*, or "CCTL exit routines" in *IMS Version 15 Exit Routines*.

2. Connect CICS to DBCTL

   After CICS has been started, the CICS operator can issue a CONNECT command to DBCTL through the CDBC transaction. This transaction is also used for disconnecting from DBCTL. The CDBI transaction provides the status of the connection.

   Related reading: For more information on these transactions, see *CICS Transaction Server for z/OS IMS Database Control Guide*.

Chapter 41. Accessing IMS databases through the ODBA interface

Open Database Access (ODBA) provides a callable interface that enables any z/OS recoverable, resource-managed z/OS address space to issue DL/I database calls to an IMS DB subsystem.

The interface provides z/OS application programs (hereafter called ODBA applications) access to full-function DL/I databases and data entry databases (DEDBs). The ODBA application and IMS must coexist on the same z/OS image.

Application programs that use the ODBA interface can use the APPLCTN macro or, if dynamic resource definition is enabled, the **CREATE PGM** command or the Program Creation user exit routine (PGMCREAT) to define the PSB names required by the ODBA applications.

The ODBA interface allows IMS DB and ODBA application programs to be developed, installed, and maintained independently. This independence provides failure isolation and resource recovery by using z/OS Resource Recovery Services (RRS).

You can achieve further failure isolation by configuring ODBA to use the CSL Open Database Manager (ODBM). ODBM can prevent an IMS abend 0113 from occurring if an ODBA application terminates unexpectedly during DL/I processing. You can configure ODBA to use ODBM by specifying the ODBMNAME and IMSPLEX parameters with the other IMS database resource adapter (DRA) interface startup parameters in the DFSxxxx0 member. The ODBA interface uses the DRA interface to communicate with IMS DB.

If you use WebSphere Application Server or WebSphere Liberty Profile Server and access IMS databases with type-2 IMS Universal Database resource adapter connectivity, you must run ODBM with Resource Recovery Services (RRS). To learn more, see IMS Universal drivers: WebSphere Application Server Liberty type-2 connections (Communications and Connections).

The ODBA interface resides in the z/OS address space and is recognized by IMS as an application region (hereafter called the z/OS application region).

Related concepts

"RRS and distributed syncpoint/protected conversations" on page 27
Regardless of whether the SYNCLVL setting is NONE, CONFIRM, or SYNCPOINT, if RRS=Y, z/OS Resource Recovery Services is the sync point manager and coordinates the update and recovery of multiple protected resources. RRS controls how and when protected resources are committed by coordinating with the resource managers, such as IMS, that have registered with RRS.

Creating the ODBA DRA start-up table

Create the ODBA DRA startup table using the DFSPRP macro.

The startup table uses the DSNAME to dynamically allocate the data set that contains the rest of the ODBA interface routines. The DDNAME is generated to allow multiple connections to IMS from the same z/OS application region. If you specify the DDNAME on the DFSPRP macro, it is ignored.

The default DSNAME is IMS.SDFSRESL. This is the default name established by the IMS definition process. Make sure this data set is APF authorized.

Note: IMS.SDFSRESL library does not contain an DRA startup table. You must generate your own table by using the DFSPRP macro. Name the load module based on the following naming conventions:

- Characters 1-3 = DFS
- Characters 4-7 = specified 4-byte ID

  The 4-byte ID should be the IMSID of the IMS system to which you will connect. However, this is not a requirement.

© Copyright IBM Corp. 1974, 2020

- Character 8 = 0 (zero)

Ensure that the DRA startup table module name is not the same as the name of an existing IMS module. To prevent accidental overlay, put the module in a load library that is accessible by the z/OS application region and not by the IMS region.

If you use a different library for your own versions, make sure that the library is APF authorized. The DRA callable interface dynamically allocates the library by using the data set name specified in the DRA startup table. The DDNAME is generated to allow multiple connections to IMS from the same z/OS application region.

To learn more about the DFSPRP macro, see DRA startup table (System Programming APIs).

Loading and running the ODBA and DRA modules in the z/OS application region

Place the following ODBA and DRA modules in the STEPLIB or JOBLIB in the z/OS application region. These modules are shipped with IMS in IMS.SDFSRESL.

DFSCDLI0
This module is bound or loaded by an application program. DFSCDLI0 also contains the ALIAS name AERTDLI.

DFSAERG0
This module is loaded by DFSCDLI0.

DFSAERM0
This module is attached by DFSAERG0 in the z/OS application region.

DFSAERA0
This module is attached by DFSAERM0 for initialization to the specified IMS DB subsystem.

Binding application programs

Bind the ODBA application programs with DFSCDLI0 (AERTDLI). As an alternative, you can issue a load and branch command passing the AIB call list in Register 1.

Related reference
The AERTDLI interface (Application Programming)

Establishing and defining security

IMS provides several options for establishing and defining security for application programs that use the ODBA interface.

The options that you select depend on the type of security environment and authorization method that you plan to use. In general, the process that IMS uses to secure PSBs involves one of the following types of security checking:

- Resource access security (RAS)

  A security check is performed by RACF to determine if the user is authorized to use the PSB. RACF determines authorization by looking at the RACF security class profile defined for the dependent region.
- APSB security

  A security check is performed to determine if the user is authorized to use the PSB.

The following table identifies the values that you need to specify to control data access for specific security implementations. The table also indicates the type of security checking that is performed for each set of specifications.

Table 141. Options for controlling data access for applications that use ODBA

Security implementation	Authorization method	ISIS= specification	ODBASE= specification	Connection security	PSB security
Resource access security	RACF	R	N		X
	Resource Access Security user exit (RASE)	C	N		X
	RACF and Resource Access Security user exit (RASE)	A	N		X
	None	0 | N	N		
APSB security	RACF	Not applicable	Y		X

To control data access for application programs that use the ODBA interface, follow the techniques discussed in the following topics to establish connection security and PSB security checking.

Related concepts

IMS security (System Administration)

Related reference

RASE: Resource Access Security user exit (DFSRAS00 and other RASE exits) (Exit Routines)

RAS security

Use the ISIS and ODBASE execution parameters to control the authorization for a z/OS application region to use a PSB.

The following table describes the actions that you need to perform to set up security when specific options are selected.

Table 142. Options for defining RAS security for applications that use ODBA

Specifications	Actions to perform
ISIS=0 | N and ODBASE=N	No action required. No PSB security checking is performed.
ISIS=R and ODBASE=N	Define the PSBs that you want protected by RACF to the IIMS or Ixxxxxxx resource class, and then define the user IDs of the dependent region that you want authorized to access the PSBs. The ODBA support for IMS will use the security environment (ACEE) passed in the dependent region's task (TCBSENV), if present, or the dependent region's address space (ASXBSENV), if the ACEE is not present at the task level.
ISIS=C and ODBASE=N	Create a Resource Access Security user exit (RASE). This routine must determine if the user is authorized to use the PSB.

Table 142. Options for defining RAS security for applications that use ODBA (continued)

Specifications	Actions to perform
ISIS=A and ODBASE=N	1. Define the PSBs that you want protected by RACF to the IIMS or Ixxxxxxx resource class, and then define the user IDs of the dependent region that you want authorized to access the PSBs. The ODBA support for IMS will use the security environment (ACEE) passed in the dependent region's task (TCBSENV), if present, or the dependent region's address space (ASXBSENV), if the ACEE is not present at the task level. 2. Create a Resource Access Security user exit (RASE). This routine must determine if the user is authorized to use the PSB. RACF is called first, and then the exit routine is called.

Defining APSB security

Use the ODBASE execution parameter to control the authorization for a user to use a PSB.

To set up APSB security when ODBASE=Y:

1. Define the PSBs that you want protected by RACF to the AIMS or Axxxxxxx general resource class (where xxxxxxx is the value specified on the RCLASS= initialization EXEC parameter).
2. Specify RCF= T | N | and ISIS= R | C at IMS system definition time.

Part 10. Open Transaction Manager Access (OTMA)

The following topics about Open Transaction Manager Access (OTMA) are for IMS system and Transaction Manager administrators responsible for installation, design, customization, operation, and recovery procedures for OTMA servers or clients.

© Copyright IBM Corp. 1974, 2020

Chapter 42. Introduction to OTMA

IMS Open Transaction Manager Access (OTMA) is a transaction-based, connectionless client/server protocol.

Though easily generalized, its implementation is specific to IMS in a z/OS sysplex environment. The domain of the protocol is restricted to the domain of the z/OS cross-system coupling facility (XCF).

OTMA addresses the problem of connecting a client to a server so that the client can support a large network, or a large number of sessions, while maintaining high performance.

Other solutions available today use network-based protocols, such as Systems Network Architecture (SNA). These protocols require a great amount of overhead because they are not transaction based.

What is OTMA?

OTMA has similarities to network protocols.

Several architectural models for networks exist. The following figure shows two. The simplified four-layer model shown on the right is often used in descriptions of UNIX networks. In the open systems interconnection (OSI) model, shown on the left, OTMA is the session layer. Both models have a Transport, Network, and Data Link layer. The OSI model also includes layers for Application, Presentation, and Session, and the simplified model includes a process layer. In the four-layer model, OTMA is the process layer.

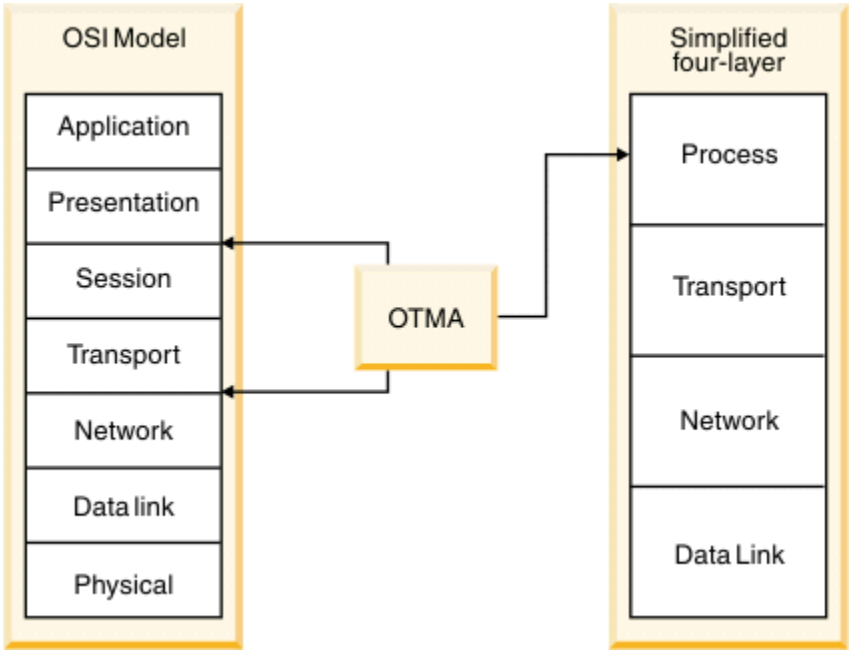


Figure 128. Network architecture models

OTMA, however, does not exactly conform to the OSI model, because OTMA can process several sessions simultaneously using a single transport connection, if the following are true:

- The z/OS cross-system coupling facility (XCF) is the transport layer.
- A session is the connection between IMS and a client.
- A client or server only creates a single XCF connection.

OTMA performs some of the basic functions of the OSI transport layer (those not performed by XCF), so it is simplest to think of OTMA as a combined session and transport layer, with the transport layer comprised of both XCF and OTMA.

© Copyright IBM Corp. 1974, 2020

Although you can think of OTMA as a session and transport layer in a network architecture model, OTMA is designed to be a high-performance comprehensive protocol that allows z/OS programs to access IMS applications.

Definitions: A *z/OS program* in this case means any z/OS application that is a member of an XCF group that includes IMS. The XCF group members that IMS communicates with are called *OTMA clients*.

By using OTMA, each client (z/OS application) can submit transactions to IMS or issue IMS commands and receive output from IMS application programs and from IMS itself.

Definition: Because IMS can communicate with, or serve, many OTMA clients, IMS is called the *server*. However, OTMA only operates in the following IMS environments:

- IMS TM and DB (the IMS DB/DC environment)
- IMS TM with Db2 for z/OS (the IMS DCCTL environment)

Related concepts

"The OTMA client" on page 821
The OTMA environment includes a server and one or more clients.

Capabilities of OTMA

OTMA capabilities include support for IMS applications, IMS and OTMA commands, resource monitoring, and more.

An OTMA client can issue most IMS commands, including both type-1 and type-2 commands, and receive responses as a result of those commands.

You can specify a level of security for each OTMA client individually. Alternatively, you can indicate that no security checking is to be done for its messages, thereby minimizing security-processing overhead.

The OTMA message flow and synchronization point protocols can be modified by an OTMA client for each transaction. In other words, the transaction-processing protocol used is not dependent on the current session.

OTMA can monitor how well IMS is processing incoming transactions and alert the client when the processing ability of the IMS system appears degraded.

OTMA can monitor transactions and cancel them if a specified transaction expiration time is exceeded before they are processed.

You can define a limit to the number of input messages from any one OTMA client that can be waiting at the same time to be processed. This prevents OTMA messages from causing an abend as a result of using too much storage. OTMA issues warning messages as the message count approaches the limit. When the message count reaches the limit, OTMA identifies a *message flood condition* and rejects any new messages from the OTMA clients until the message count drops or a new limit is set.

You can stop incoming transactions and commands from individual OTMA clients without stopping the connection or affecting any transactions or commands that IMS is already processing.

OTMA supports sending transaction messages from IMS application program running in the dependent regions of a local IMS system across a TCP/IP network to another IMS system for processing. IMS Connect is required to manage the TCP/IP connection.

OTMA also supports callout requests from IMS application programs running in IMS dependent regions. Callout is invoked with the DL/I ICAL call. Callout requests allow IMS application programs to request data or services from servers that are outside of the IMS installation. Depending on whether the callout request is processed synchronously or asynchronously, OTMA support for callout requests differs.

OTMA can route callout messages to IMS Connect and its clients, such as IMS Enterprise Suite SOAP Gateway and IMS TM Resource Adapter, and to IBM MQ. IMS uses OTMA internally (whether or not OTMA is enabled) to route synchronous program switch requests to IMS applications.

The IMS `/DISPLAY TRANSACTION` command output is in the form of an OTMA message returned to the client in the application-data section of the message prefix.

OTMA-initiated transactions are identified to z/OS Workload Manager using the OTMA transaction-pipe name, which identifies the logical connection between IMS and OTMA.

Existing IMS application programs that have previously not been used with OTMA can run without modification and interact with OTMA clients. APPC/IMS application programs that use IMS SET0 calls might need some modification.

Related concepts

"OTMA support for callout requests" on page 849
IMS application programs running in IMS dependent regions can call out through the Open Transaction Manager Access (OTMA) component of IMS to servers that are outside of the IMS installation to request data or services. The synchronous callout interface can also be used to request services from another IMS application with a synchronous program switch.

"Specifying an expiration time for transactions to OTMA" on page 803
You can specify an expiration time for a transaction to reduce processing costs by preventing IMS from processing transactions that the client can no longer use.

"Monitoring system resources with OTMA" on page 799
OTMA monitors IMS system resources that are used to process OTMA transactions and notifies OTMA clients about how well the IMS system is processing OTMA transactions.

Related reference

"OTMA restrictions and requirements" on page 791
A number of general restrictions and requirements apply to OTMA.

Benefits of using OTMA

IMS applications that do not use OTMA do not need to be modified to use OTMA, unless they use SETO calls, in which case, you might need to modify them to use OTMA.

The SET0 call applies to APPC/IMS and SPOOL/API processing.

For each OTMA-originated transaction, the SET0 call returns a status code. You can bracket the SET0 call with an INQY call if necessary.

Full-duplex processing provides an environment in which transactions and output messages are sent and processed in parallel.

You can implement IMS device support outside IMS. You can also implement device support for your IMS subsystem that is different from what IMS provides, or enable device support that IMS does not provide. The following figure illustrates how IMS can communicate with a device, shown here as a workstation, using device support implemented within an OTMA client. IMS device support using Virtual Telecommunications Access Method (VTAM) is shown for comparison.

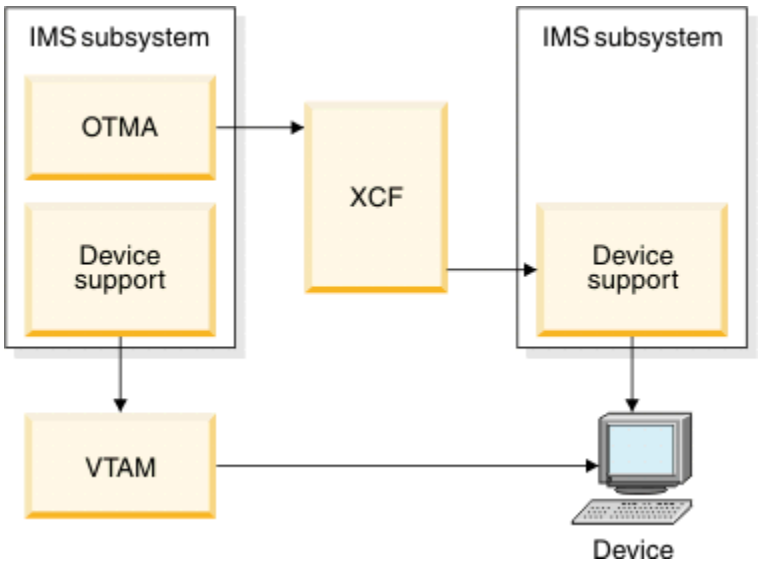


Figure 129. IMS communicates with a device using device support implemented within an OTMA client

Flow-control and transaction-processing attributes are dynamically bound to the transaction.

Clients have high-performance access to IMS:

- OTMA uses the z/OS cross-system coupling facility application programming interface (API).
- OTMA does not use VTAM and IMS device-dependent support.

Transactions based on different protocols (that is, that have different processing requirements such as being recoverable or irrecoverable) can be associated with a single transaction pipe.

You can connect up to 255 clients to the OTMA group.

Messages can be extended using the user-data section of the message prefix, allowing additional user information to be sent with the transaction.

User information and transaction pipe name are included within the messages themselves.

Different clients can specify the same transaction pipe names, instead of needing to use uniquely named resources.

You do not need to use networking architectures, such as SNA (Systems Network Architecture).

Related concepts

"Using transaction pipes with OTMA" on page 762
An IMS transaction represents a request for IMS to do some work. Many transactions also require a response after IMS has completed the work. So, each transaction has a source (the requester) and often a destination (for the response).

Related reference

"Using DL/I calls in an OTMA environment" on page 816
Certain DL/I calls have special considerations when used with OTMA.

Advantages of the OTMA protocol

OTMA treats transactions as data objects that have attributes independent of application-, session-, or transport-layer considerations. OTMA is, in effect, a transaction layer, independent of other layers.

As a unique layer, OTMA offers flexibility, simplicity, and performance that other solutions do not offer. This section outlines the transaction-specific services that OTMA provides the client.

Grouping of transactions using transaction pipes.

Security options (for example, the client can verify security or let the server verify the user ID).

Dynamically-bound flow control and processing. The client can decide how transaction requests and responses are to be processed by the server.

The ability for the client to query the server for transactions that the server supports.

Treating transactions as objects. The client can include any pertinent user data with the transaction, and allow that data to stay with all messages generated by the transaction.

The ability for the client to specify a client token with each transaction to correlate input with output.

The ability for the client to control transaction processing performed by the server, in terms of:

- The client can set a different level of security than that of the OTMA group to which it belongs.
- To improve performance, the client can eliminate the security-checking that the server performs on the messages it delivers.
- Transaction grouping, using the transaction-pipe token.
- OTMA clients that use synclevel=confirm or synclevel=synchpt can specify the timeout value for send-then-commit messages.

Client routing. An OTMA destination descriptor can be easily coded in the DFSYDTx PROCLIB member to reroute an output message that is inserted to an alternate PCB to any OTMA client or to IMS. Alternatively, an IMS exit routine can coded to reroute output messages that are inserted to an alternate PCB.

Architected command output. The client can use the IMS `/DISPLAY TRANSACTION` command to query the server's transaction attributes and receive the reply in a structured format. Therefore, the need for automated operator scripting to control processing is reduced.

Unlike APPC, when using message flow through transaction pipes, no concept exists of a session that contains the flow-control parameters for all transactions and associated output data for the session.

How IMS messages flow in an OTMA environment

The key to message flow for OTMA is the *transaction pipe*, the logical connection between the server and the OTMA client.

An OTMA client includes the transaction-pipe name in the message-control information section of the message prefix for the input message. IMS then associates application output for an OTMA client with a specific transaction pipe.

Related concepts

“Using transaction pipes with OTMA” on page 762
An IMS transaction represents a request for IMS to do some work. Many transactions also require a response after IMS has completed the work. So, each transaction has a source (the requester) and often a destination (for the response).

Basic OTMA message flow

The basic message flow for OTMA clients is relatively simple.

The basic message flow is:

1. The client submits a transaction or command to IMS.
2. IMS accepts IMS transactions as input from any client.

   The IMS transaction code is specified in the application-data section of the input message.

   If the client is submitting an IMS command, the command is included in the application-data section of the input message.
3. The input message is processed.

   An IMS transaction is enqueued to the appropriate application program using an IMS scheduler message block (SMB).

An IMS command is processed by IMS. The output is sent to the client synchronously or asynchronously, depending on the type of request.

4. Application output is sent to the client.

   Generation of output and commit are coordinated based on the commit mode specified in the state-data section of the message prefix for the input message.

   The application output is enqueued to a dynamically created IMS transaction-pipe structure (specific to that client) before being sent to the client.

   For an OTMA-submitted transaction, IOPCB output is returned to the OTMA client. By default, all alternate PCB output is also sent to the OTMA client. You can change this by coding either an OTMA destination descriptor in the OTMA DFSYDTx PROCLIB member or by coding the OTMA Destination Resolution user exit (OTMAYPRX) or the client's OTMA User Data Formatting user routine (DFSYDRU0). You can also use these exit routines to route alternate PCB output from non-OTMA-submitted transactions to OTMA clients.

   IMS delivers segmented messages in order, even though the z/OS cross-system coupling facility (XCF) does not guarantee sequential delivery of messages.

The following figure shows an example of the message flow in an OTMA environment. Two clients are shown side by side in the example; they can be a TCP/IP client, a IBM MQ Queue Manager client, or a client of any other network type. Message flow starts with the client, goes through the XCF group, and to IMS. Within the IMS address space, a control region contains OTMA; the message flow ends at a transaction-pipe. The IMS application program issues a Get Unique (GU) call in the dependent region.

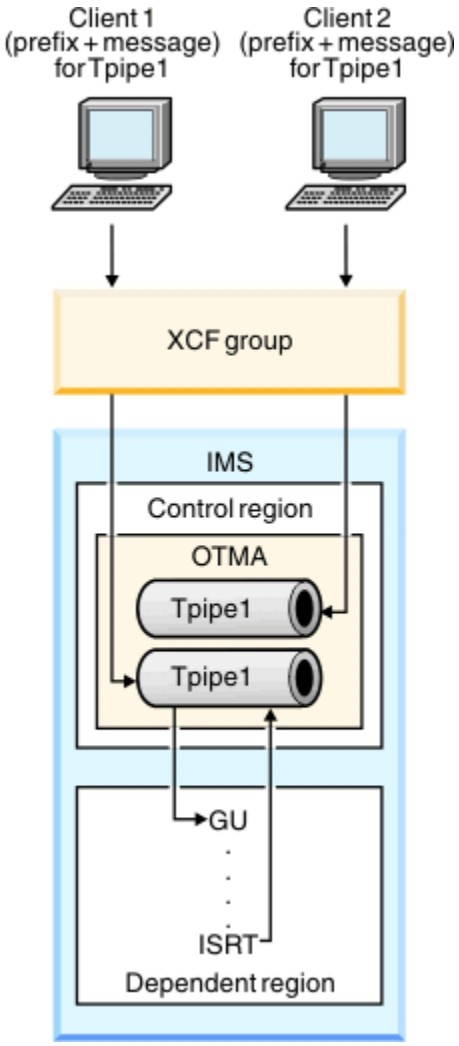


Figure 130. IMS message flow in an OTMA environment

In the preceding figure:

1. The message prefix is always attached to the input transaction, even in the case of segmented input. This prefix contains important information, such as the transaction-pipe name and the client token.

   A client application program can send several transactions specifying the same transaction-pipe name. The client token must always be present in the prefix, so that the client application program knows how to process the IMS output it receives.
2. OTMA clients do not need to predefine transaction pipes. Two different clients can use the same transaction-pipe name. Although many clients can use the same transaction-pipe name, each transaction pipe is unique. `client 1` and `client 2` both use `tpipe1`, yet each is a unique transaction pipe.

   A client can create and use as many transaction pipes as it needs.
3. The transaction-pipe structure is created dynamically when OTMA receives output and is used as an anchor for the application output.
4. The IMS application program has no knowledge of the OTMA message prefix when it issues the GU call.

IMS supports a full-duplex message flow for a client/server session. The client can instead request a half-duplex message flow, but this flow must be implemented and managed by the client itself:

- A correlator token in the state-data section of the message prefix can be used to uniquely identify a transaction. IMS maintains this field in the message prefix for a transaction.
- The client can set the response-requested flag in the message-control information section of the message prefix to receive a response for a message.
- Any unsolicited output from IMS is easily identified by a client, because the message prefix specifies only the transaction-pipe name. The client can ask IMS to discard the output.

  Unsolicited output should not interfere with half-duplex processing. That is, the client must be prepared for full-duplex flows while still maintaining a half-duplex flow on a user-token level. Contention should not be an error condition.

OTMA IMS-to-IMS TCP/IP communications flow

An IMS application program that is running locally in an IMS dependent region can send transaction messages across a TCP/IP network to a remote IMS system for processing.

The flow of a transaction message that is sent to another IMS system across a TCP/IP network is illustrated by the following steps.

1. An IMS application program that is running in the local IMS system issues a CHNG call with the name of an OTMA destination descriptor specified as the destination for the transaction message and then issues an ISRT ALTPCB call to send the transaction message. As an alternative, instead of using an OTMA destination descriptor, you can use an OTMA User Data Formatting exit routine (DFSYDRU0) in the local IMS system to route ALTPCB output to remote IMS systems.
2. IMS routes the transaction message to a local IMS Connect instance based on the information specified in the OTMA destination descriptor. If a super member group is active locally, IMS distributes output transactions to up to 8 IMS Connect instances in the super member group by using a round-robin algorithm.
3. Using commit-then-send (CM0) and a send-only with acknowledgment protocol, IMS Connect sends the transaction message to a remote IMS Connect instance on a TCP/IP connection that is identified by a client ID generated by the local IMS Connect instance.
4. The remote IMS Connect instance sends the message to OTMA in the remote IMS system.
5. In the remote IMS system, OTMA returns an acknowledgement (ACK) to IMS Connect and queues the message as an IMS transaction.
6. An IMS application program running in the remote IMS system processes the transaction.
7. Any output generated by the IMS application program in the remote IMS system is queued to the tpipe hold queue

The following figure illustrates the flow.

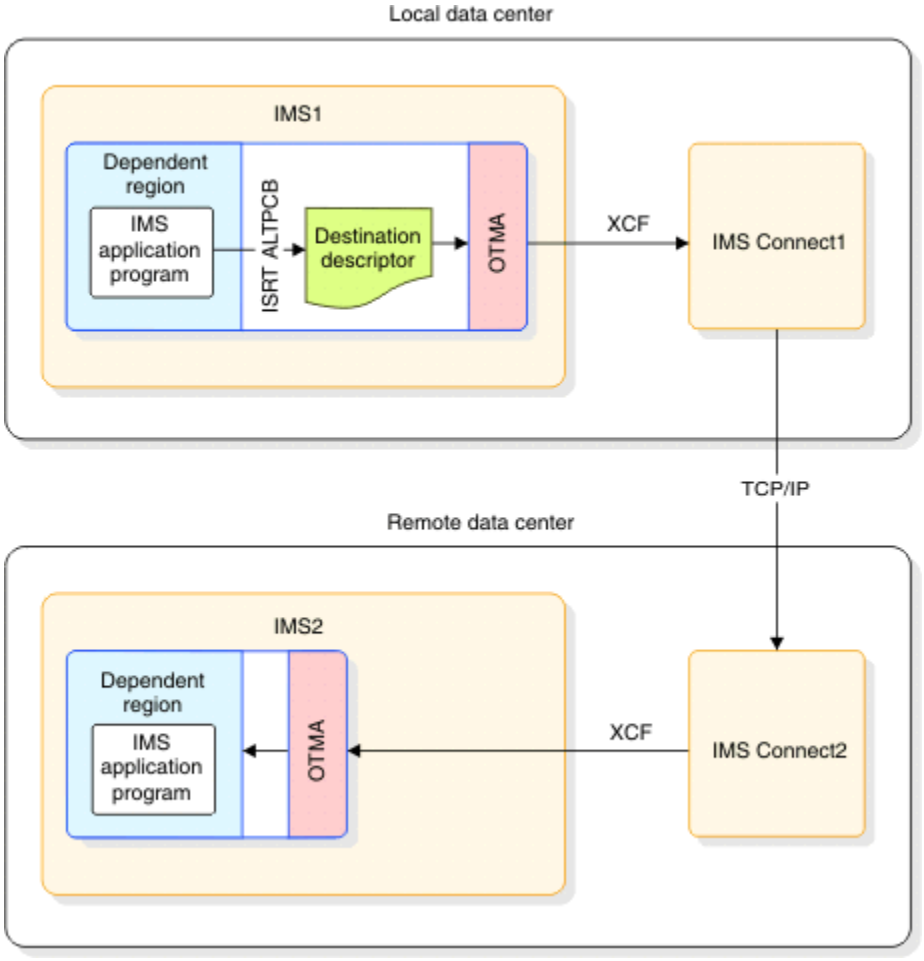


Figure 131. IMS-to-IMS TCP/IP communications flow for an OTMA transaction message

Related concepts

"Defining OTMA destination descriptors" on page 776
You can create, delete, and update OTMA destination descriptors either by using the online type-2 IMS commands or by coding them directly in the DFSYDTx member of the IMS.PROCLIB data set.

"IMS Connect support for IMS-to-IMS TCP/IP communications" on page 150
IMS Connect manages the TCP/IP connections and protocols for IMS systems that communicate with each other across a TCP/IP network.

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

Sample commit-then-send transaction processing flows

The following figure shows a non-OTMA environment: a secondary logical unit type 2 (SLU 2) device communicates with IMS using VTAM and IMS device support (DDMs). The transactions are enqueued to the IMS message queues. Transaction output is returned to the SLU 2 device.

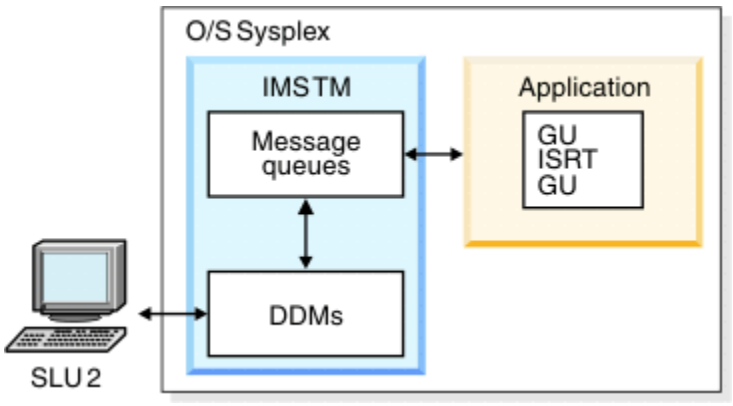


Figure 132. Standard SLU 2 transaction flow

The following figure shows the same transaction flow in an OTMA environment. The transaction still comes from a SLU 2 device, but the device communicates with IMS using an OTMA client, through an z/OS cross-system coupling facility (XCF) group, rather than VTAM.

The figure only shows the input flow, which begins with the SLU 2 device, goes to the OTMA client, through the XCF group, and ends at the OTMA server. The transaction is placed on the message queue, and the application issues get unique, insert, and get unique calls. Output follows the same path, in reverse. Of course, if a client is to send output to the SLU 2 device, the SLU 2 device must be defined to the client, and the client must be able to drive that device.

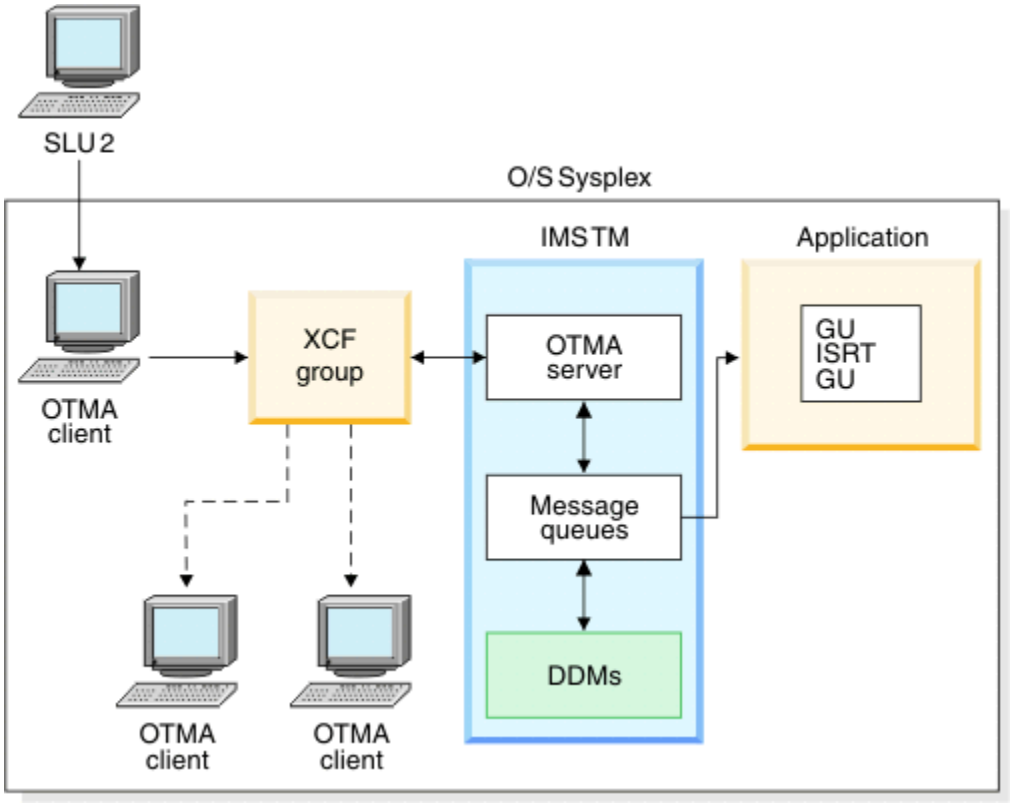


Figure 133. SLU 2 transaction flow using OTMA

It might seem that the OTMA flow is more complex, and for a SLU 2 device, perhaps it is. But you can use OTMA to allow any type of device to communicate with IMS, not just VTAM-supported devices. An OTMA client can also act as a gateway for another network, such as a TCP/IP network.

Using transaction pipes with OTMA

An IMS transaction represents a request for IMS to do some work. Many transactions also require a response after IMS has completed the work. So, each transaction has a source (the requester) and often a destination (for the response).

IMS uses the concept of a logical terminal (LTERM) to ensure that responses are associated with the correct requesters. An LTERM uses a queue where the transaction output is kept before it is returned to the requester.

Definition: A *transaction pipe* (tpipe) is a logical connection between a client and the server. It is analogous to an IMS logical terminal (LTERM). For each LTERM, IMS maintains a connection between the queue and the physical node that receives the output. OTMA does not use an LTERM but still must maintain a connection between the client and IMS. This connection is the transaction pipe, or tpipe.

Transaction pipes enable a client to associate its transactions with a transaction-pipe name. IMS uses the transaction-pipe name to associate all input and output with a particular client. The association between the transaction output and its ultimate destination (for example, a user at a terminal or a printer) is not made within IMS (as is the case with LTERMs), but is the responsibility of the client.

By using a transaction pipe, IMS does not know anything about the actual user of the transaction, often a user of the client application. Because IMS does not know anything about the actual user, the client has complete control over the output of transactions.

OTMA's use of transaction pipes provides:

- **Flexibility**

  Many transaction outputs can flow through the same transaction pipe.
- **Performance**

  Transaction pipes give the client the ability to specify and distinguish transactions based on their message-flow control and synchronization.
- **Resynchronization between a client and IMS**

  Transaction pipes can be either synchronized or non-synchronized. For a synchronized transaction pipe, all output messages are serialized through a single process, and sequence numbers can be assigned to messages. By logging these serialized messages, IMS and the client can resynchronize in the event of an outage.

  No resynchronization is required for a non-synchronized transaction pipe.
- **Object orientation**

  A transaction can be thought of as an object because OTMA keeps the transaction message information (such as user data and transaction-pipe name) within the message.

IMS removes transaction pipes after they have been idle for three consecutive system checkpoints, except in the following circumstances:

- Commit-then-send messages are queued on the tpipe or the tpipe hold queue.
- The tpipe is stopped.
- A trace is set on the tpipe.
- The tpipe is a synchronized tpipe, such as a tpipe used by MQSeries® for commit-then-send input transactions.
- The tpipe is in a WAIT state for a resume tpipe request that specified either the AUTO or the SINGLE-WAIT options.
- The tpipe is in an MCP state, which indicates that the tpipe is running in a shared queues environment and might have output messages on the global queue.

Tip: If no messages are queued to the TPIPE but the MCP status is displayed for the TPIPE so that the tpipe cannot be removed, issue the `/DISPLAY TMEMBER tmembername TPIPE tpipename QCNT` command or the `/DISPLAY TMEMBER tmembername QCNT` command to reset the MCP status.

- The tpipe is being scanned by IMS.

You can use the **/DISPLAY TMEMBER TPIPE** command to see whether a tpipe cannot be removed by IMS because one of the circumstances in the preceding list is true for the tpipe.

The following figure illustrates how transaction pipes fit in an OTMA client/server environment. As shown in the figure, transaction-pipe structures reside in the OTMA layer only for the server. z/OS cross-system coupling facility, which resides in the transport layer, can be thought of as an interprocess communication layer, because it provides communication between the client process and the server process.

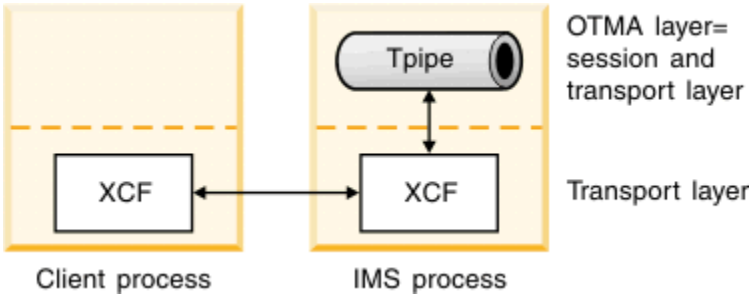


Figure 134. How transaction pipes fit in an OTMA client/server environment

Related reference

/DISPLAY TMEMBER command (Commands)

Differences in transaction pipes

IMS LTERMs and UNIX pipes both provide a one-way flow for message traffic. An OTMA transaction pipe provides a two-way flow.

The concept of a transaction pipe is applicable to any protocol. In a general way, the transaction pipe replaces the IMS LTERM because:

- Processing is full duplex.
- Multiple flow-control mechanisms are possible.
- The logical output entity (in other words, the LTERM) is dissociated from the node of the actual user.
- The transaction pipe is implemented as a protocol rather than as an API, which facilitates a client/server architecture.
- The transaction pipe sets up a data-control mechanism independent of session characteristics, and is therefore transaction specific.

Message flow using transaction pipes

The flow control of transactions is handled by the client.

The client dynamically binds flow-control parameters to the transaction by querying the transaction attributes in the server. Transaction pipes are not usually associated with flow control (except for synchronized transaction pipes using half-duplex processing).

The following figure shows the basic message flow between a client and a server, using the z/OS cross-system coupling facility (XCF). The order of processing is:

1. The client sends a transaction as input to the server (IMS).
2. The server returns transaction output messages to the client.

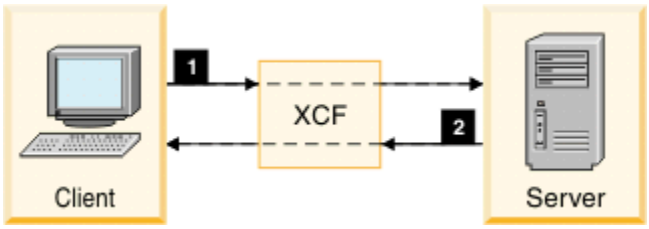


Figure 135. Basic transaction-pipe message flow

Within the server, the input transaction and the output messages are organized and synchronized using IMS queues, as shown in the following figure. The figure illustrates a commit-then-send transaction flow for a non-Fast Path environment.

The order of processing is:

1. The client sends a transaction to the server, and the server enqueues the transaction on a message queue.
2. The transaction is submitted to an application program for processing.
3. The application program prepares any output for the transaction and commits the output during sync-point processing.
4. The output is returned to the client.

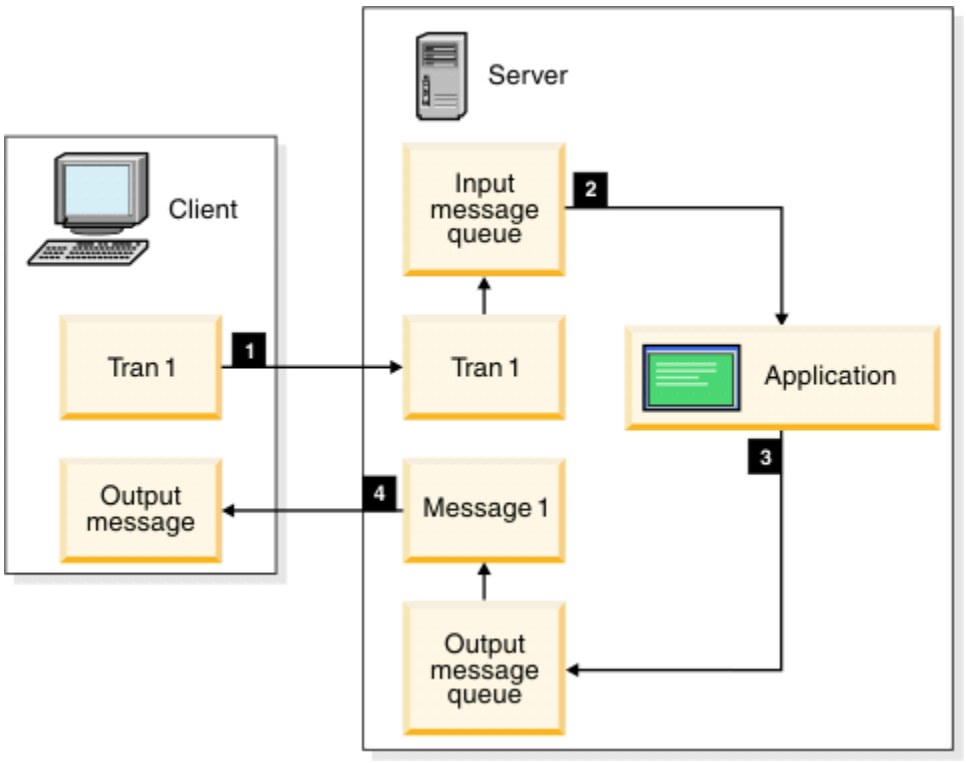


Figure 136. Use of queues in the transaction-pipe message flow

In a full-duplex environment, transactions and output messages are being sent and processed in parallel, as shown in the following figure. This parallelism can be maximized by creating a process for every transaction and output message. The order of processing is:

1. The client sends a transaction (`Tran 1`) to the server, and the server's transaction pipe enqueues the transaction.
2. The transaction (`Tran 1`) is submitted to an application program for processing.
3. The application program enqueues any output (`Message 1`) for the transaction (`Tran 1`).

4. The client sends a second transaction (`Tran  2`) to the server and the server's transaction pipe enqueues the transaction.
5. The second transaction (`Tran  2`) is submitted to an application program for processing.
6. The output (`Message  1`) for `Tran  1` is returned to the client.
7. The application program enqueues the output (`Message  2`) for the second transaction (`Tran  2`).
8. The output (`Message  2`) for `Tran  2` is returned to the client.

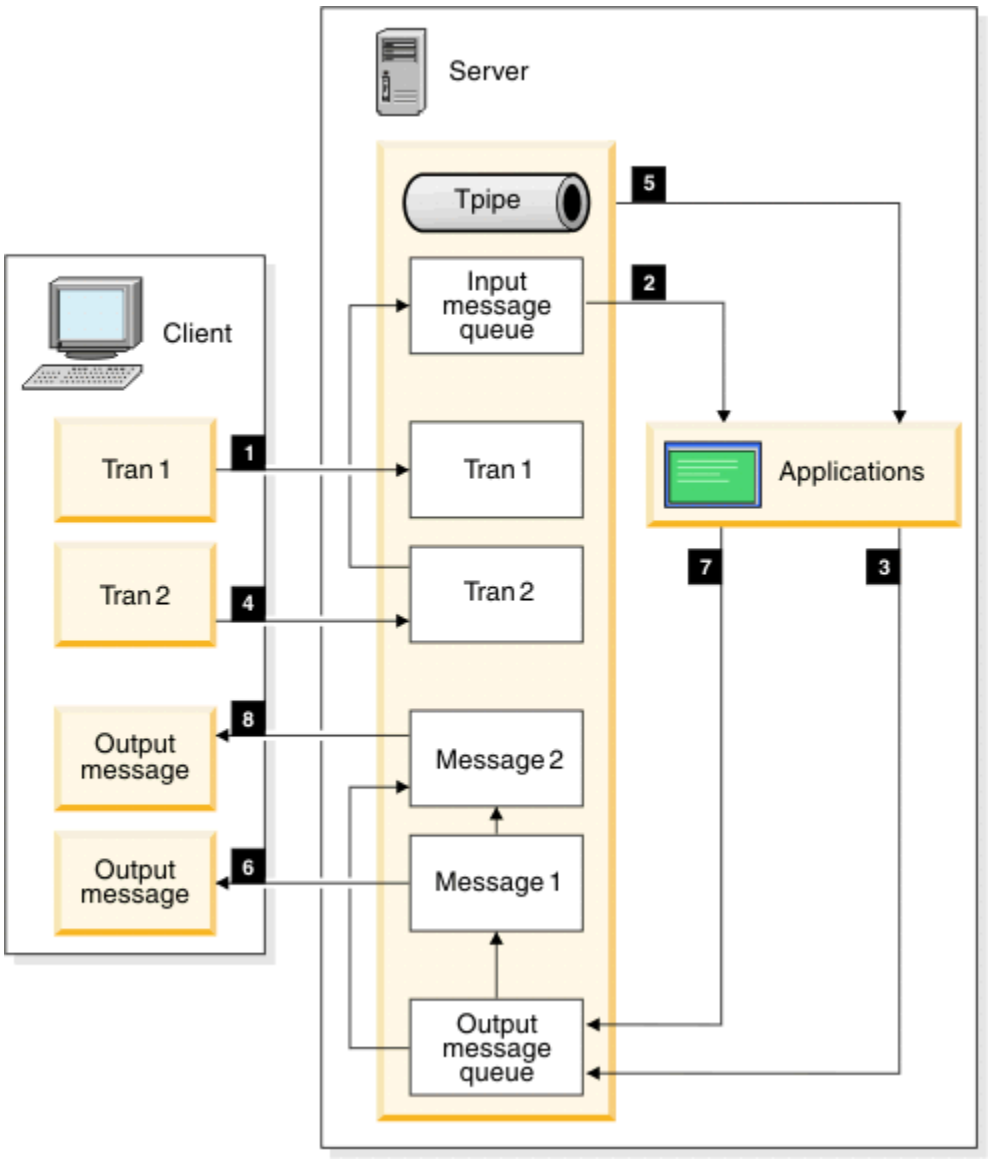


Figure 137. Transaction-pipe flow in full-duplex environment

Q: Does specifying a transaction pipe as synchronized make the communication flow half-duplex?

A: No. Transaction pipes are always full-duplex.

Whether the communication flow is actually half-duplex depends on the client. For a synchronized transaction pipe, IMS processes all output messages in the order received. No new messages are sent for the transaction until IMS has received an ACK message for the previous message. A NAK message causes IMS to halt all output processing for that transaction.

While this output processing is taking place, the client could be sending new input transaction messages to IMS on that synchronized transaction pipe. If the client coordinates the sending of transactions with the receipt of IMS output, the client can effect half-duplex processing.

Related concepts

"OTMA commit processing" on page 824
OTMA can control how IMS commits transactions: they can be either commit-then-send or send-then-commit.

Chapter 43. Enabling and using OTMA

The following topics describe IMS tasks related to the OTMA environment, as well as how IMS operates in an OTMA environment.

Enabling OTMA

To enable IMS to use OTMA, specify the z/OS cross-system coupling facility (XCF) group name and IMS OTMA member name during system definition.

OTMA is installed with IMS TM. The IMS INSTALL/IVP Dialog is not used to install OTMA.

To start OTMA, you can use the **OTMA=Y** startup parameter in the IMS procedure during IMS system definition or, after an IMS restart, issue the type-1 command **/START OTMA**.

You can define various OTMA attributes during IMS system definition, including z/OS cross-system coupling facility (XCF) names, startup behavior, security, and so on.

Related reference

DFSPBxxx member of the IMS PROCLIB data set (System Definition)

Parameter descriptions for IMS procedures (System Definition)

IMS procedure (System Definition)

DCC procedure (System Definition)

Summary of the OTMA configuration parameters

All the following OTMA configuration parameters can be specified in the DFSPB*xxx* member of the IMS.PROCLIB data set and either the IMS or DCC startup procedures.

GRNAME=
Defines the name of the XCF group that IMS creates and joins for OTMA communications.

OTMA=
Specifies whether OTMA starts during IMS startup and whether the **/START OTMA** command is recoverable across warm and emergency restarts.

OTMAASY=
For send-then commit messages, controls the synchronous or asynchronous scheduling of transaction originating from a program-to-program switch.

OTMANM=
Specifies the XCF member name that identifies OTMA within the XCF group.

OTMAMD=
Determines whether the member override field in the parameter list of the OTMA Destination Resolution user exit (OTMAYPRX) can be used to specify a different XCF member name for transactions invoked from an OTMA client.

OTMASP=
For IBM MQ, specifies whether a synchronized tpipe is used for OTMA output.

OTMASE=
Specifies the type of OTMA RACF security that you want to use, if any.

Related tasks

“OTMA security” on page 806
Security for OTMA is enforced by RACF or a similar security product. The following topics describe implementing security for OTMA and OTMA clients by using RACF.

Related reference

DFSPBxxx member of the IMS PROCLIB data set (System Definition)

© Copyright IBM Corp. 1974, 2020

Parameter descriptions for IMS procedures (System Definition)
IMS procedure (System Definition)
DCC procedure (System Definition)

Defining the XCF group name

OTMA communications require XCF. Both IMS and the OTMA clients must join the same the XCF group.

You specify the XCF group name for OTMA communications on the GRNAME parameter in either the DFSPBxxx PROCLIB member or the IMS control region JCL. If a valid XCF group name is not specified, OTMA cannot be started.

When the GRNAME parameter is specified, IMS joins the XCF group only when OTMA is started. If you have configured OTMA to start when IMS starts, IMS joins the XCF group during startup. If OTMA is not configured to start when IMS starts, IMS joins the XCF group only after OTMA is started by the **/START OTMA** command.

If you specify GRNAME= and OTMA is started, you can use the **/DISPLAY OTMA** command to display the XCF status. You are not required to define any XCF information.

If you use RACF for security, the `IMSXCF.group.member` (client member name) must be defined in the RACF FACILITY class.

Related reference
Parameter descriptions for IMS procedures (System Definition)

Defining the OTMA XCF member name

XCF requires an XCF member name for each member of an XCF group.

You can specify an XCF member name for OTMA by using the OTMANM parameter. If you do not specify the OTMANM parameter, OTMA uses the APPLID of the IMS system as the OTMA XCF member name.

The OTMANM parameter is not used in Extended Recovery Facility (XRF) installations. Instead, IMS uses a USERVAR name as the XCFmember name. The USERVAR name is specified in either the IMS control region JCL, the DFSPB*xxx* PROCLIB member, or the DFSHSB*xx* PROCLIB member.

Recommendation: Do not change the group name or the IMS member name during an IMS emergency or normal restart.

Related reference
Parameter descriptions for IMS procedures (System Definition)

Defining when OTMA starts up

You can configure OTMA to start when IMS starts or you can start OTMA after IMS is running.

Prerequisite: Before you can start OTMA, you must first specify the GRNAME execution parameter. The GRNAME parameter identifies the XCF group that OTMA uses for communications.

- To start OTMA during IMS startup, specify OTMA=Y in either the DFSPBxxx PROCLIB member or the IMS control region JCL.
- To start OTMA after IMS is running, specify OTMA=N, OTMA=M, or omit the OTMA parameter completely. After IMS is running, issue the **/START OTMA** command.

  Note: The **/START OTMA** command is recoverable. That is, if OTMA was started by the **/START OTMA** command, IMS disregards an OTMA=N specification and restarts OTMA during warm and emergency restarts, even if OTMA=N or the OTMA parameter is not specified.
- To prevent IMS from restarting OTMA during a warm or emergency restart after the **/START OTMA** command has been issued, specify OTMA=M. When OTMA=M is specified, IMS does not recover the **/START OTMA** command during warm or emergency restarts.

Related reference

Parameter descriptions for IMS procedures (System Definition)

Defining the level of OTMA security checking

If you use a security product such as RACF, you can specify different levels of security checking by using the OTMASE parameter.

The value specified in the OTMASE parameter can be overridden when IMS is running by issuing the /SECURE OTMA command.

The levels of security checking include:

CHECK (C)
IMS commands are checked against the CIMS class. IMS transactions are checked against the TIMS class.

FULL (F)
The same type of security as CHECK, but additional checking is performed against dependent regions. F is the default value for the OTMASE parameter.

JOIN (J)
Only OTMA client bid requests are checked by using the RACF Facility class IMSXCF.*xcfgroup.member* profile, if it exists. No calls to RACF are made for IMS transactions and commands.

NONE (N)
OTMA RACF security is NONE. No calls to RACF are made.

PROFILE (P)
Each OTMA message defines the level of security checking to be done.

After the OTMA client connection is authorized, the transaction or command security checking will be performed based on the security setting in the LUY_RACF_OPT of the message prefix which is set using the OTMA security level. For OTMA security level JOIN (J), the security setting in the message will be NONE for transaction and command processing.

Related reference

Parameter descriptions for IMS procedures (System Definition)

/SECURE command (Commands)

OTMA tpipe support for parallel processing of multiple active RESUME TPIPE requests

When MULTIRTP=Y is specified in an OTMA client descriptor, the OTMA tpipes that are associated with the OTMA client can support multiple active resume tpipe requests in parallel, unless the MULTIRTP specification is overridden by the client.

By default, OTMA creates tpipes that support only a single active **RESUME TPIPE** request and queues any additional **RESUME TPIPE** requests until the active **RESUME TPIPE** request terminates. Supporting only a single active **RESUME TPIPE** request provides more control over the order in which the output messages from OTMA are processed.

Enabling support for the parallel processing of multiple active **RESUME TPIPE** requests can significantly increase the throughput of an OTMA tpipe for output messages, particularly those for synchronous or asynchronous callout requests, and can significantly improve failover protection for OTMA tpipes. Although OTMA sends the output messages in the order in which they are created from the IMS application programs, differences in the performance of both the network connections and the IMS Connect client application programs make predicting the order in which the output is acknowledged and processed unpredictable.

When an OTMA tpipe supports multiple active RESUME TPIPE requests, OTMA clients can pull output messages from the tpipe by using multiple active RESUME TPIPE requests, which the tpipe processes in parallel. If the processing for any one RESUME TPIPE request becomes impaired, OTMA continues

to deliver the output messages on the tpipe through the other active RESUME TPIPE requests, which prevents the RESUME TPIPE request, or the tpipe itself, from becoming a bottleneck for output messages from IMS.

OTMA tpipe support for multiple active RESUME TPIPE requests also improves failover protection for OTMA tpipes by eliminating the need to switch to a back up OTMA client if the active OTMA client terminates. When an OTMA tpipe supports multiple active RESUME TPIPE requests from multiple clients, if any one of the OTMA clients fail or lose their connection, the others can continue processing the output from the tpipe without any lapse in processing.

Support for multiple active RESUME TPIPE requests can also decrease the complexity and cost of routing output from multiple IMS application programs through OTMA tpipes to the same final destination. Without the parallel processing of RESUME TPIPE requests, to achieve optimum performance, as well as to avoid a potential bottleneck, the output from the IMS application programs is typically routed through multiple OTMA destination descriptors or OTMA tpipes; however, such a configuration usually requires some combination of unique coding in each IMS application program, in multiple OTMA destination descriptors, and in the OTMA clients. With support for multiple active RESUME TPIPE requests, you can realize similar performance benefits by routing the output from the multiple application programs through a single OTMA destination descriptor and a single OTMA tpipe. Multiple OTMA clients can then retrieve the output by issuing the same RESUME TPIPE requests with the OTMA tpipe name specified as an alternate client ID.

For diagnostic purposes, when support for multiple active RESUME TPIPE requests is enabled, the RESUME TPIPE token can be used to correlate each RESUME TPIPE request with the client that issued it. The RESUME TPIPE token, as well as the ID of any tpipe to which undelivered output is rerouted, can be displayed by issuing the existing commands that support OTMA and IMS Connect.

Related concepts

“OTMA client descriptors” on page 773
Use OTMA client descriptors to provide information about a specific OTMA client to IMS. OTMA client descriptor entries are identified in the DFSYDTx PROCLIB member by an M in column one of the descriptor entry.

Related reference

OTMA client descriptor syntax and parameters (System Definition)

Enabling parallel processing of RESUME TPIPE requests

OTMA tpipe support for the parallel processing of RESUME TPIPE requests can be enabled in several different ways.

Support for parallel processing of multiple RESUME TPIPE requests can be enabled in IMS or by OTMA clients that use the RESUME TPIPE request.

- To set the IMS system default for OTMA tpipe support for parallel processing of RESUME TPIPE requests, specify MULTIRTP=Y in the OTMA system client descriptor, DFSOTMA, in the DFSYDTx PROCLIB member
- To specify support for a single OTMA client, specify MULTIRTP=Y in the OTMA client descriptor for that OTMA client. The MULTIRTP specification for specific clients overrides the IMS system default.
- OTMA clients can enable MULTIRTP support in their client bid requests by specifying X'80' in the TMAMHFG3 field at byte offset 64 in the state data for client bids section of the OTMA header on client bid requests. Specifications made in a client bid requests by an OTMA client override specifications made on OTMA client descriptors in IMS.

OTMA super members and parallel RESUME TPIPE support

OTMA super members can support multiple active **RESUME TPIPE** requests when MULTIRTP=Y is specified by an OTMA client, such as IMS Connect.

Enabling MULTIRTP support in a super member group can improve both the throughput of the super member group during normal operations and the failover support of the super member group if a member of the group stops for any reason.

When MULTIRTP support is enabled, messages on the super member tpipe hold queue are sent immediately on any available connection with an active **RESUME TPIPE** request without waiting for an acknowledgement to the previously sent message. If your OTMA clients require the output that is received from the super member to be processed in chronological order, do not enable MULTIRTP support.

When MULTIRTP support is not enabled, only one **RESUME TPIPE** is active at a time and the messages are not sent until an acknowledgement for the previous message is received.

All OTMA clients that connect to the same super member in the same IMS system must use the same MULTIRTP setting for parallel **RESUME TPIPE** request support.

The first client to connect to the super member defines the MULTIRTP setting for the super member group. If a client that connects to the super member after the first client has a MULTIRTP value that is different, the client bid request is rejected with a NAK and sense code X'0037'.

Related tasks

“Sharing asynchronous commit-then-send output: the OTMA super member function” on page 847
Hold-queue-capable OTMA clients, such as IMS Connect, can share asynchronous commit-then-send (CM0) output messages by enabling the OTMA super member function. The OTMA super member function is specifically designed to support multiple instances of IMS Connect in a z/OS Sysplex Distributor environment.

Enabling or changing MULTIRTP support in an OTMA super member

Support for parallel active **RESUME TPIPE** requests (MULTIRTP) by an OTMA super member tpipe queue is determined by the MULTIRTP value in the client bid request of the first client to connect to the super member group.

To change the MULTIRTP value of an existing super member group, you must terminate all client connections and re-establish the client connections after the clients are configured with the intended MULTIRTP value.

1. If you are changing the MULTIRTP value of an existing super member group, terminate all data store connections to the super member group.

   For connections from IMS Connect, you can stop the data store connections by specifying any of the following commands:

   - Type-2 **UPDATE IMSCON** TYPE(DATASTORE) NAME(*datastore_name*) STOP(COMM)
   - WTOR STOPDS *datastore_name*
   - z/OS UPDATE DATASTORE NAME(*datastore_name*) STOP(COMM)
2. Configure the OTMA clients with the MULTIRTP value that you need.

   For IMS Connect, you can specify the MULTIRTP parameter in either the IMS Connect system configuration or in the definition of individual data store connections.
3. Start the OTMA client connections to the super member.

   For connections from IMS Connect, you can start the data store connections by specifying any of the following commands:

   - Type-2 **UPDATE IMSCON** TYPE(DATASTORE) NAME(*datastore_name*) START(COMM)
   - WTOR STARTDS *datastore_name*
   - z/OS UPDATE DATASTORE NAME(*datastore_name*) START(COMM)

When the data store connections are started, the OTMA super member takes the MULTIRTP value of the first client to connect and processes the RESUME TPIPE requests from the OTMA clients according to the new MULTIRTP value.

If any client attempts to connect to the super member group with a MULTIRTP value that is different from the MULTIRTP value that is established in the super member group, is rejected by a NAK message with sense code X'0037'.

Specifying synchronized tpipes for IBM MQ

An OTMA client, such as IBM MQ, can require that an OTMA tpipe is synchronized for ALTPCB output.

You can specify synchronized tpipes in two ways: by setting the output flag in the OTMA User Data Formatting exit routine (DFSYDRU0) or by specifying OTMASP=Y. If the only reason you code the DFSYDRU0 exit routine is to set the synchronized output flag, you can use the OTMASP parameter instead.

By default, OTMA creates non-synchronized tpipes for the OTMA output.

If your organization uses both the DFSYDRU0 exit routine and the OTMASP parameter to control the tpipe that gets created for OTMA output, the following table shows when synchronized tpipes are created.

Table 143. Tpipes created when both OTMASP parameter and DFYDRU0 exit used

DFSYDRU0 is set to...	If OTMASP=Y, result is...	If OTMASP=N, result is...
Create synchronized tpipe	Synchronized tpipe	Synchronized tpipe
Create non-synchronized tpipe	Synchronized tpipe	Non-synchronized tpipe

Related reference

Parameter descriptions for IMS procedures (System Definition)

OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits) (Exit Routines)

Enabling OTMAYPRX member name override for OTMA clients

The parameter list for the OTMA Destination Resolution user exit (OTMAYPRX) contains a member override field that allows you to route transaction received from non-OTMA LTERMs to OTMA clients.

By default, the member override field cannot be used to route transactions received from OTMA clients.

- To use the member override field in the OTMAYPRX parameter list for transactions received from both OTMA clients and non-OTMA LTERMs, specify OTMAMD=Y.

Specifying asynchronous delivery of program-to-program switch output messages

If you are using send-then-commit (CM1) messages that initiate multiple program-to-program switches, to ensure that only the appropriate output is returned to the OTMA client in synchronous CM1 mode, specify OTMAASY=Y.

When OTMAASY=Y is specified, if an input transaction triggers multiple program-to-program switches, only the response transaction is scheduled by OTMA synchronously.

Non-response transactions that originate from a program-to-program switch are scheduled asynchronously, regardless of whether they are returned to the client before the expected synchronous output. IMS does not issue DFS2082 messages for transactions that are scheduled asynchronously.

When transactions are scheduled asynchronously, IMS does not issue DFS2082 messages.

When OTMAASY=N is specified, or the OTMAASY parameter is omitted, if any output from a program-to-program switch is returned before the expected synchronous output, errors can occur.

When OTMAASY=S is specified, if an input transaction triggers multiple program-to-program switches, only the first transaction originating from a program-to-program switch that was performed through ISRT to a ALTPCB (if non-Express) is scheduled by OTMA synchronously.

Related reference

Parameter descriptions for IMS procedures (System Definition)

OTMA descriptors

OTMA provides two types of descriptors: an OTMA client descriptor and an OTMA destination descriptor.

The OTMA client descriptor specifies certain characteristics of an OTMA client, such as its DFSYDRU0 exit routine, its send-then-commit message time out values, or its message flood threshold value.

The OTMA destination descriptor defines destination attributes for:

- OTMA ALTPCB messages that are sent to IMS Connect for delivery to a remote IMS system by way of a TCP/IP connection with another IMS Connect instance.
- OTMA ALTPCB output sent to either IMS Connect or non-OTMA destinations such as an SNA terminal or printer.
- Callout requests from IMS applications programs that are sent to IMS Connect and that are ultimately destined for any of the following:
  - An enterprise Java bean (EJB) or message-driven bean (MDB) running under a web application server, such as WebSphere Application Server.
  - A Web service.
  - A user-written IMS Connect client.

Both OTMA client descriptors and OTMA destination descriptors are specified as entries in the DFSYDTx PROCLIB member.

OTMA destination descriptors can also be created and updated online by using the following type-2 IMS commands:

- CREATE OTMADESC
- DELETE OTMADESC
- QUERY OTMADESC
- UPDATE OTMADESC

Related reference

DFSYDTx member of the IMS PROCLIB data set (System Definition)

OTMA client descriptors

Use OTMA client descriptors to provide information about a specific OTMA client to IMS. OTMA client descriptor entries are identified in the DFSYDTx PROCLIB member by an M in column one of the descriptor entry.

The information in the OTMA client descriptor helps IMS manage messages from the OTMA client. The information included in an OTMA client descriptor can also be specified in an OTMA client-bid request or, in some cases, in a /START OTMA command. Information provided by these alternate methods might override the information provided in the OTMA client descriptor.

Using an OTMA client descriptor, you can specify to OTMA and IMS the following attributes of an OTMA client:

- The name of the OTMA Destination Resolution exit routine that the OTMA client uses.
- The timeout value that OTMA should use when waiting for an ACK or NAK response from the OTMA client for both commit-then-send messages and send-then-commit messages that use synclevel=confirm or synclevel=syncpt.

- The maximum number of input messages from the OTMA client that can be processing in an IMS system before triggering a message flood condition.

OTMA client descriptors are optional; however, they can be particularly useful for routing output to an OTMA client before the OTMA client has actually connected to OTMA. When the OTMA client finally does connect, the specifications in the client-bid request override the specifications made in the OTMA client descriptor.

The following defaults apply if you do not specify any of the parameters of the OTMA client descriptor:

- IMS uses the default OTMA Destination Resolution exit routine, DFSYDRU0, unless the client-bid request specifies a different exit routine name.
- For the acknowledgment timeout value, IMS uses a default of two minutes unless a different timeout value is provided in either the client-bid request or the TIMEOUT parameter of the **/START TMEMBER** command.
- For message flood detection, IMS uses a default threshold of 5000 messages, unless a different value is provided in either the client-bid request or the INPUT parameter of the **/START TMEMBER** command.

OTMA client descriptors are built during IMS initialization. The descriptors are included in DFSYDTx members of IMS.PROCLIB. The "x" on DFSYDTx is the IMS nucleus suffix. If you have multiple clients at the same IMS system, list each client name on a separate line. Ensure that columns 3-18 are different for each client.

By default, you can define a maximum of 254 client descriptors. If you do not use the DFSOTMA descriptor, you can specify a maximum of 255 client descriptors.

You can change the maximum number of client descriptors by specifying the MDESCMAX parameter in the DFSOTMA descriptor.

Related reference

DFSYDTx member of the IMS PROCLIB data set (System Definition)

OTMA destination descriptors

Use OTMA destination descriptors to define destinations for messages that are routed through OTMA.

OTMA destination descriptors provide a simpler method for describing destinations than using the OTMA Destination Resolution user exit (OTMAYPRX) and the OTMA User Data Formatting exit routine (DFSYDRU0).

OTMA destination types include:

TYPE=IMSCON
IMS Connect destinations

TYPE=IMSTRAN
Other IMS application programs via synchronous program switch

TYPE=MQSERIES
IBM MQ

TYPE=NONOTMA
Non-OTMA destinations

When OTMA destination descriptors are used to describe a destination, IMS does not call the OTMAYPRX user exit and DFSYDRU0 exit routine. You can override this behavior for descriptors with TYPE=IMSCON, TYPE=NONOTMA, or TYPE=MQSERIES by specifying EXIT=YES. In that case, the exits are called to process the message and determine whether the descriptor routing information must be modified.

IMS Connect descriptors

OTMA destination descriptors support the routing of callout requests from IMS application programs to external data or service providers through IMS Connect. IMS Connect routes callout requests through one of the following IMS Connect clients:

- IMS TM Resource Adapter
- IMS Enterprise Suite SOAP Gateway
- User-written IMS Connect clients

For any of the following types of messages, define your destination type as IMS Connect (TYPE=IMSCON):

- Output for IMS Connect clients
- Callout requests from IMS application programs
- Transaction messages routed to another IMS system for processing through IMS-to-IMS TCP/IP communications

The destination attributes that you can specify in the descriptor differ depending on the type of destination.

For IMS Connect destinations, you can specify the following destination attributes:

- An OTMA tmember name
- An OTMA super member name
- A tpipe name
- For transaction messages destined for a remote IMS system via an IMS-to-IMS TCP/IP connection:
  - The name of a connection to a remote IMS Connect instance that is defined in an RMTIMSCON configuration statement of the local IMS Connect instance
  - The IMS ID of a remote IMS system, as defined in a DATASTORE configuration statement of a remote IMS Connect instance
  - Optionally, the transaction code that the remote IMS system schedules to process the ALTPCB message
  - Optionally, the user ID that the remote IMS system uses to perform transaction authorization for the ALTPCB message
- For IMS Connect XML conversion support for SOAP Gateway, an XML adapter name and an XML converter name.
- A timeout interval for synchronous callout requests that are made by IMS application programs that issue the DL/I ICAL call. If a response is not received in the specified amount of time, the request times out and the IMS dependent region is freed. If the timeout value specified on the OTMA destination descriptor differs from the timeout value specified by the IMS application on DL/I ICAL call, OTMA uses the smaller of the two values.

For IMS Connect destinations, the OTMA destination descriptor provides a layer of abstraction between the application program and the tpipes. This layer of abstraction provides flexibility in the routing of IMS Connect output. For example, the destination name specified in the descriptor can be explicit or generalized using a mask character. Also, the tpipe name specified in the descriptor can be an individual tpipe name, a super member name, or left blank, in which case the specified destination name is used as the tpipe name.

IBM MQ descriptors

OTMA can route asynchronous callout requests to IBM MQ with a TYPE=MQSERIES descriptor and the following information:

- An OTMA tmember name
- An OTMA super member name
- A tpipe name
- The MQMD_REPLYTOQ and MQMD_REPLYTOQMGR values for your IBM MQ application
- Optionally, an override to call the OTMA routing exits (DFSYPRX0 and DFSYDRU0)
- Optionally, other IBM MQ values for the MQ message descriptor structure

IMS synchronous program switch descriptors

You can route a message directly to another IMS application program by creating a descriptor with TYPE=IMSTRAN. You can optionally specify other attributes for the descriptor:

- An OTMA tmember and tpipe that late response messages from switched applications are routed to
- An override to call the IMS user exit (DFSCMUX0) for late response messages
- A check that IMS can perform to determine if the target application replies to the IOPCB

Non OTMA descriptors

Non OTMA destination descriptors support the routing of information to destinations such as SNA terminals/devices and printers. You can define your destination type as non-OTMA by specifying TYPE=NONOTMA.

Related reference

DFSYDTx member of the IMS PROCLIB data set (System Definition)

Defining OTMA destination descriptors

You can create, delete, and update OTMA destination descriptors either by using the online type-2 IMS commands or by coding them directly in the DFSYDTx member of the IMS.PROCLIB data set.

By default, you can define a maximum of 510 destination descriptors. You can change the maximum by specifying the DDESCMAX parameter in the DFSOTMA descriptor.

The type-2 commands for administering OTMA destination descriptors include:

- **CREATE OTMADESC**
- **DELETE OTMADESC**
- **QUERY OTMADESC**
- **UPDATE OTMADESC**

Changes made to the OTMA destination descriptors by any of the above type-2 commands become effective immediately and are recorded in both the IMS system logs as x'221B' log records and in the IMS checkpoint logs as x'4035' log records. The changes made by the type-2 commands are not stored in the DFSYDTx PROCLIB member and are retained only across warm starts and emergency restarts of IMS. The changes made by the type-2 commands are not retained across cold starts of IMS.

If you use the type-2 commands to create, modify, or delete OTMA destination descriptors, the changes override any existing definitions for the same descriptors that are in the DFSYDTx until the next cold start.

Changes made in the DFSYDTx member require a cold start of IMS to take effect, because IMS reads the DFSYDTx member only during a cold start.

In the DFSYDTx member, the OTMA destination descriptors are distinguished from the OTMA client descriptors, which are also stored in the DFSYDTx member, by a D in the first column of the descriptor entry.

You can update and delete OTMA destination descriptors that are coded in the DFSYDTx member with the type-2 commands; however the changes are not reflected in the DFSYDTx member. The modified or deleted descriptors are recorded in the logs, and the log records override the descriptor statements that are stored in the DFSYDTx member.

The **QUERY OTMADESC** command returns information about the OTMA destination descriptors as it is currently recorded in the logs.

Related reference

CREATE OTMADESC command (Commands)
DELETE OTMADESC command (Commands)
QUERY OTMADESC command (Commands)
UPDATE OTMADESC command (Commands)

OTMA destination descriptor syntax and parameters (System Definition)

Masking names of OTMA destination descriptors

When specifying a destination name in an OTMA destination descriptor, you can generalize destination names by using an asterisk as a wildcard character or as a mask of the final characters in the destination name field.

For example, if you specify OTICON* as the destination name, the descriptor applies to any output that specifies a destination name that begins with OTICON, such as OTICON01 and OTICON02.

The OTMA destination descriptors can be listed in any order in the DFSYDTx PROCLIB member; however, when IMS searches for a descriptor to route output, IMS searches the descriptors in order from the most specific destination name to the most general. That is, IMS always reads specific destination names before destination names that include an asterisk.

If a descriptor specifies a masked destination name, such as OTICON*, but does not specify a tpipe name, the output must specify the actual tpipe name in full. By default, if no tpipe name is specified on the descriptor, the destination name is used as the tpipe name. If the destination name is masked, then IMS must read the destination name specified by the output to determine the specific tpipe to use.

If you specify a tpipe name on a descriptor that uses a masked destination name, all output sent to destinations that match the masked destination name is routed to the same tpipe.

DFSOTMA descriptor

Use the optional DFSOTMA descriptor to define global and default attributes, limits, and types of support that apply to all OTMA clients.

The DFSOTMA descriptor is an OTMA type-M descriptor that sets system defaults and global parameters for all OTMA clients that connect to this IMS system.

The DFSOTMA descriptor does not support all of the parameters that you can specify on an OTMA client descriptor. Parameters that are not supported, such as DRU= and T/O=, are ignored if they are specified in the DFSOTMA client descriptor.

The global values that you can specify for all OTMA clients include:

- Message flood protection
- Global tpipe limit for all the OTMA members or clients
- Support for multiple active resume tpipe requests
- Save area prefix allocation limits
- The maximum number of OTMA descriptors
- The maximum number of RACF user IDs that can have cached accessor environment elements (ACEEs) stored in subpool 249.
- Passing the input LTERM override name to IBM Workload Manager

Because the DFSOTMA descriptor is a type-M descriptor, when the DFSOTMA descriptor is used, the maximum number of OTMA client descriptors that you can define is one less, because OTMA client descriptors are also type-M descriptors.

The DFSOTMA descriptor is always loaded from the DFSYDT*x* PROCLIB member regardless of the IMS restart type. There are no checkpoint log records for the DFSOTMA descriptor.

Related tasks

“OTMA ACEE flood control” on page 800
The OTMA accessor environment element (ACEE) flood control function prevents virtual storage in the IMS control region from running out.

Related reference

DFSOTMA descriptor syntax and parameters (System Definition)

Changing the limits on OTMA descriptors

By default, IMS limits the number of OTMA descriptors that you can define to 255 type-M descriptors and 510 type-D descriptors. You can change these limits.

OTMA descriptors are stored in the extended common storage area (ECSA). Limiting the number of OTMA descriptors that you can define ensures that the descriptors do not use too much storage.

Usually, the amount of storage that OTMA uses for OTMA descriptors at any point in time is less than the defined maximum limit because OTMA allocates only enough ECSA storage for the OTMA descriptors that are currently defined.

To change the maximum number of OTMA descriptors that can be defined:

1. If your DFSYDT*x* member already contains a DFSOTMA descriptor, make sure it is one of the first 255 type-M descriptors listed in the DFSYDT*x* member of the IMS PROCLIB data set.

   Recommendation: Code the DFSOTMA descriptor as the first entry in the DFSYDT*x* member so that it is easy to find and there is no risk of it being listed after the 255th type-M descriptor.
2. If your DFSYDT*x* member does not contain a DFSOTMA descriptor, create one. It must be one of the first 255 type-M descriptors listed in the DFSYDT*x* member of the IMS PROCLIB data set.

   Recommendation: Code the DFSOTMA descriptor as the first entry in the DFSYDT*x* member so that it is easy to find and there is no risk of it being listed after the 255th type-M descriptor.
3. In the DFSOTMA descriptor, specify the new maximum number for the type-D or type-M descriptor.
   - For type-D descriptors (destination descriptors), specify the new maximum number on the DDESCMAX parameter
   - For type-M descriptors (client descriptors and the DFSOTMA descriptor), specify the new maximum number on the MDESCMAX parameter
4. Restart IMS to add the new or updated DFSOTMA descriptor to the online IMS system

OTMA support for IMS-to-IMS communications

You can send OTMA messages from a local IMS system to a remote IMS system by using IMS-to-IMS TCP/IP communications.

To send a message to a remote IMS system, you define the remote IMS system as an OTMA destination in either an OTMA destination descriptor or an OTMA User Data Formatting exit routine (DFSYDRU0). IMS application programs running in IMS dependent regions can then insert messages to an ALTPCB with the OTMA destination name specified. OTMA queues the messages to a tpipe for IMS Connect, which delivers the message to the remote IMS system by way of the TCP/IP network.

You can use the OTMA client descriptor to set an ACK timeout interval that determines how long OTMA waits for an ACK or NAK response to a transaction message sent to a remote system.

OTMA messages destined for remote IMS systems flow one way only. To return responses that are generated by the remote IMS system to the local IMS system, the return path must be defined separately in the remote IMS system and the response must be returned as a separate transaction.

Local and remote instances of IMS Connect manage the TCP/IP connection between the local IMS system and the remote destination IMS system. The TCP/IP connection between the two IMS systems is defined in IMS Connect by using the RMTIMSCON configuration statement in addition to the other required IMS Connect configuration statements.

Use persistent connections for OTMA IMS-to-IMS TCP/IP connections to minimize the risk of an excessive number of tpipes accumulating on the remote IMS system. IMS cleans up idle tpipes unless output is queued to them.

You specify connection persistence to the local IMS Connect instance by specifying PERSISTENT=Y on the RMTIMSCON statement that defines the connection to the remote IMS Connect instance.

Related concepts

"IMS Connect support for IMS-to-IMS TCP/IP communications" on page 150

IMS Connect manages the TCP/IP connections and protocols for IMS systems that communicate with each other across a TCP/IP network.

Related tasks

IMS-to-IMS TCP/IP connections (System Definition)

Related reference

OTMA destination descriptor syntax and parameters (System Definition)

Super member support for IMS-to-IMS communications

You can use the OTMA super member function to distribute messages sent to a remote IMS system across multiple local instances of IMS Connect.

OTMA distributes IMS-to-IMS messages in turn to each IMS Connect instance in the super member group by using a round-robin distribution algorithm. If an IMS Connect instance joins or leaves the super member group, OTMA dynamically updates the super member round robin list.

OTMA supports up to eight tmembers in a super member group that is used for IMS-to-IMS TCP/IP communications. OTMA counts each connection defined by an IMS Connect DATASTORE configuration statement as a tmember. The tmember limit can be reached by eight or fewer IMS Connect instances, depending on how many DATASTORE connections each IMS Connect instance establishes with the same super member group.

If more than eight tmembers are defined to use a super member group, OTMA uses only the first eight to join the super member group.

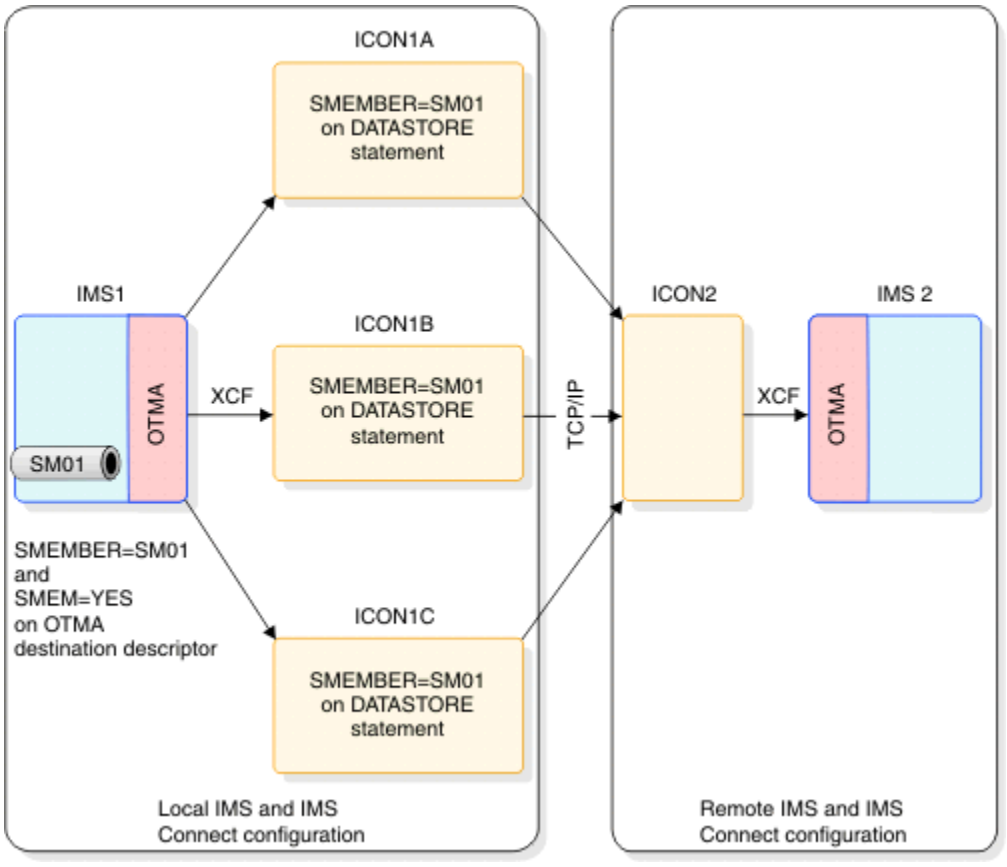


Figure 138. Using a super member group with OTMA IMS-to-IMS TCP/IP connections

Related tasks

Defining an IMS Connect super member group for OTMA IMS-to-IMS TCP/IP connections (System Definition)

Specifying a remote transaction code

OTMA messages sent to a remote IMS system can include a transaction code to schedule a transaction to process the message in the remote IMS system.

The transaction code can be specified by the sending application program, an OTMA destination descriptor, or both.

If a transaction code is specified by the application program, it is included in the first 8 bytes of the data portion of the outgoing message.

If a transaction code is specified by an OTMA destination descriptor, IMS Connect increases the length of the message by 8 bytes and inserts the transaction code into the message between the length field and the data section. If the transaction code is less than 8 bytes, the remainder is padded with blanks.

The specification of a transaction code by an OTMA destination descriptor is independent of the specification of a transaction code by a sending application program. If both the application program and the OTMA destination descriptor specify a transaction code, both are included in the outgoing message and the remote IMS system must be able to process both.

For example, if an application specifies a transaction code, the format of the message is LLZZMSGDATA, where the LLZZ section contains the length of the message and the MSGDATA section contains both the transaction code and the message data. If an OTMA destination descriptor also specifies a transaction code, the message length is increased by 8 bytes and the format of the message becomes LLZZTRANCODEMSGDATA, where TRANCODE contains the transaction code specified on the OTMA destination descriptor.

- To specify a transaction code in an OTMA destination descriptor, specify the RMTTRAN parameter.

Related concepts

"OTMA destination descriptors" on page 774
Use OTMA destination descriptors to define destinations for messages that are routed through OTMA.

Related reference

OTMA destination descriptor syntax and parameters (System Definition)

Format of messages sent to a remote IMS system

The OTMA remote ALTPCB output message created by the local IMS application must follow the standard format for IMS transaction messages.

The following example shows the standard format for IMS transaction messages:

```
LLZZ | TRANCODE | DATA
```

The following table describes the parts of the format:

Table 144. Transaction message format

Field Name	Field Length	Description
LL	2	The length of the message, including the LL and ZZ fields.
ZZ	2	Binary zeros.

Table 144. Transaction message format (continued)

Field Name	Field Length	Description
TRANCODE	8	A transaction to be scheduled in the remote IMS system. If a transaction code is specified on the RMTTRAN parameter of the OTMA destination descriptor, IMS Connect inserts the transaction code here.
DATA	Variable	The data to be processed by the IMS application program on the remote IMS system. If the sending application program specifies a transaction code, it is included here in the first 8 bytes of the DATA section.

Related reference

Input message format and contents (Application Programming)

OTMA-supported exit routines

Several exit routines delivered with IMS are specifically for OTMA. OTMA also supports several other exit routines delivered with IMS that are not specific to OTMA.

The following exit routines support OTMA:

- OTMA User Data Formatting exit routine (DFSYDRU0)
- OTMA Input/Output Edit user exit (OTMAIOED)

  Restriction: The OTMAIOED user exit is not supported for synchronous callout request messages, including those received by OTMA from IMS application programs that issue the DL/I ICAL call.
- OTMA Destination Resolution user exit (OTMAYPRX)
- OTMA Resume Tpipe Security user exit (OTMARTUX). You can use this exit routine to secure messages on the asynchronous hold queue.

In addition, the following exit routines are supported by OTMA:

- Command Authorization (DFSCCMD0)
- TM and MSC Message Routing and Control user exit (DFSMSCE0)
- Queue Space Notification (DFSQSPC0)
- Transaction Authorization (DFSCTRN0)

Related tasks

“Securing messages on the asynchronous hold queue” on page 813
You can protect messages on asynchronous hold queues from unauthorized use of the RESUME TPIPE call by using either RACF, the OTMA Resume TPIPE Security user exit (OTMARTUX), or both.

Related reference

OTMA User Data Formatting exit routine (DFSYDRU0) (Exit Routines)

OTMA Input/Output Edit user exit (DFSYIOE0 and other OTMAIOED type exits) (Exit Routines)

OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits) (Exit Routines)

OTMARTUX: OTMA Resume TPIPE Security user exit (DFSYRTUX and other OTMARTUX type exits) (Exit Routines)

Using the OTMAYPRX user exit and DFSYDRU0 exit routine to determine destination

Transaction pipe names can be the same as IMS LTERM names or APPC/IMS TP names.

To clarify whether a destination is for OTMA, IMS provides OTMA exit routines that can specify where IMS should look to resolve the destination names:

- The OTMA Destination Resolution user exit (OTMAYPRX)
- The OTMA User Data Formatting exit routine (DFSYDRU0)

In an IMS subsystem, you can have many DFSYDRU0 exit routines, but only a single OTMAYPRX user exit.

The exit routines cannot change the actual destination name.

Determining the destination for an OTMA message requires two phases. In each phase, an OTMA exit routine can be called:

Phase 1
The Destination Resolution user exit (OTMAYPRX) is called to determine the initial destination for the output.

The user exit can determine whether the message should be directed to an OTMA client or to IMS TM for processing. The exit routine cannot determine the final destination (because insufficient parameters are passed to it).

Phase 2
The DFSYDRU0 exit routine is called to determine the final destination for the output. Each client can specify a separate DFSYDRU0 exit routine.

The name of the DFSYDRU0 exit is determined by the user or an OTMA client. Each client can have its own dedicated DFSYDRU0 exit. To view the name of the DFSYDRU0 exit routine associated with an OTMA client, issue the `/DISPLAY TMEMBER` command.

Both of these exit routines receive control when an IMS application program issues an `ISRT` call to an alternate program communication block (PCB), or issues `CHNG` or `PURG` calls. But if the destination is the name of an IMS scheduler message block (SMB), the DFSYDRU0 exit routine does not receive control. The following figure illustrates the two phases of message destination determination.

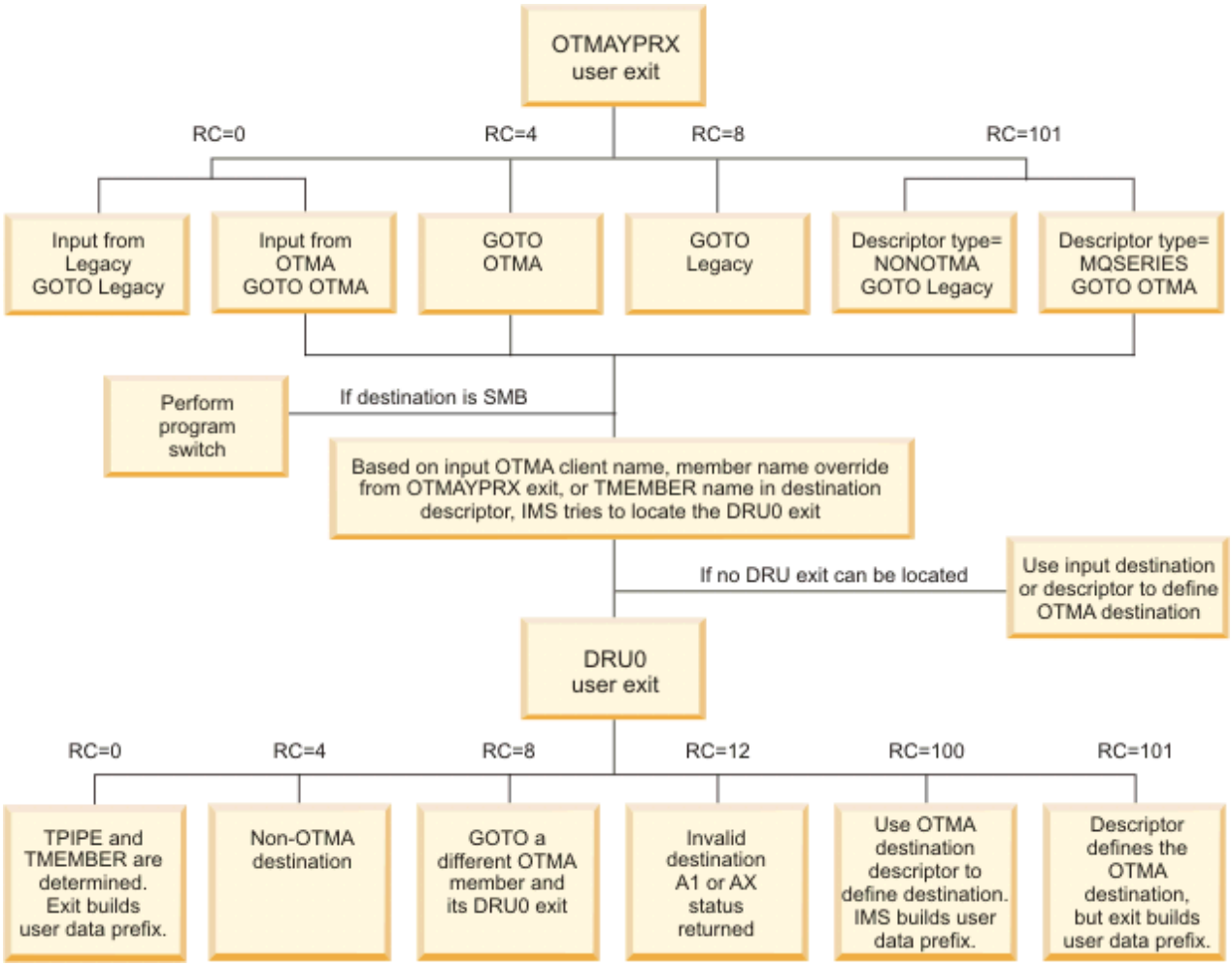


Figure 139. How the OTMAYPRX and DFSYDRU0 OTMA user exits determine message destination

Recommendations:

- The destination name that is specified at offset +8 in the input parameter list of the DFSYDRU0 exit can come from the original CHNG call, the OTMA destination descriptor, or a TPIPE override that is set by the OTMAYPRX exit routine. If the input flag X'40' is specified at offset +27 in the input parameter list of the DFSYDRU0 exit, the destination name from the original CHNG call is included in the input parameter list of the DFSYDRU0 exit routine at the offset +104.
- Because of the potential conflict with the SMB name, OTMA clients should avoid using a transaction pipe name as either the transaction name or the routing key.
- The OTMAYPRX and DFSYDRU0 OTMA user exits should be the same for the front-end and back-end IMS systems within a shared queues group. If the exit routines are different for one or more back-end IMS systems, asynchronous output might be sent to different destinations, depending on which back-end IMS system processed the input.

  To ensure prompt delivery of the output, enable OTMA on every back-end IMS system in the shared queues group. If a back-end IMS system does not have OTMA enabled, any asynchronous OTMA output that is inserted into an alternate PCB is simply queued and not delivered until the operator issues a `/STA  OTMA` command.
- Specifying OTMAMD=Y in the IMS PROCLIB member DFSPBxxx can direct your OTMA message from the OTMAYPRX user exit to a different DFSYDRU0 exit routine without rerouting.
- Specifying OTMASP=Y in the IMS PROCLIB member DFSPB*xxx* always creates a SYNC tpipe for the ALT-PCB output message.

Note: The SCD and PST addresses are available in the input parameter for both OTMA user exits. The address of the first segment of the output message is not passed to either user exit.

Related reference

OTMA User Data Formatting exit routine (DFSYDRU0) (Exit Routines)

OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits) (Exit Routines)

Administering IMS for OTMA

The following topics describe IMS administration considerations for OTMA.

IMS conversations and OTMA

OTMA-to-IMS conversations are send-then-commit and are nonrecoverable.

IMS creates a unique conversation ID for each transaction pipe that is saved in the Server token field in the OTMA state-data section of the message. This ID must be passed to IMS with all subsequent client input for the conversation using that transaction pipe.

The server-state flag is set to `Conversational State` in the state-data section of the message prefix when IMS acknowledges the transaction. This flag must be set for all subsequent client input during the conversation.

An IMS conversation can be terminated by issuing the `/EXIT` command. Alternatively, if a client wants to terminate the IMS conversation, the client can send a message with the commit-confirmation flag set in the message-control information section of the message prefix (indicating a deallocate flow). IMS then terminates the IMS conversation. A deallocate flow must specify `Conversational State` for the server-state flag in the state-data section of the message prefix; it must also set the server token field.

MSC and OTMA transactions

You can use IMS Multiple Systems Coupling (MSC) with OTMA transactions.

MSC processing in an OTMA environment is similar to MSC processing in an APPC/IMS environment:

- A client sends a transaction to IMS. If the transaction is defined as remote for that IMS system, it is sent to a remote IMS system for processing. If the transaction is defined as a local transaction and performs a message switch to another IMS system, the switched message is sent to that remote IMS system for processing.
- Output from the remote application program is returned to the originating IMS.
- IMS recognizes that the data is OTMA data and uses the transaction pipe to send the data to the client.

If the remote application inserts to an alternate PCB that is a remote destination, the data is not routed to an OTMA destination. The remote destination does not route the output message to the OTMA client, even though the message has a prefix. If the message is to be properly routed back to the original client, the remote IMS must insert to a remote transaction. That transaction (at the original IMS site) must then send the message to the OTMA client using an alternate PCB and a Prerouting exit routine.

You can use MSC with OTMA in a shared-queues environment, as long as the MSC link exists in the front-end IMS subsystem that is connected to the OTMA client.

Fast Path and OTMA transactions

Fast Path transactions must run as send-then-commit transactions.

Any parameters with the OTMA transaction that contradict this commit mode cause the transaction to be rejected. Existing Fast Path application programs can run with OTMA if the client-entered transaction is properly defined.

IMS restart processing and OTMA

If an IMS subsystem connected to an z/OS cross-system coupling facility group must be restarted, IMS reconnects to the group during restart, but all clients must send new client-bid requests to IMS.

XRF processing and OTMA

XRF supports OTMA.

Recommendation: The active and alternate IMS subsystems should be in the same z/OS cross-system coupling facility (XCF) group. If they are not in the same XCF group, after an XRF takeover, the client must connect to the new XCF group. If they are in the same XCF group, after an XRF takeover, the client is automatically reconnected to the XCF group.

The active IMS joins the XCF group specifying the IMS USERVAR name as its member name. The alternate IMS joins the group only after an XRF takeover, and connects using the same USERVAR name as the member name. If the active and XRF alternate subsystems are in the same XCF group, during XRF takeover, the new active IMS:

- Ensures that the old IMS member is disconnected
- Rejoins the group
- Begins delivering any queued output destined for an OTMA client, following client-bid processing for that client

IMS XRF does not track the **/START OTMA** and **/STOP OTMA** commands. The IMS procedures for both the active and alternate IMS should both specify OTMA=Y in order to ensure that IMS automatically rejoins the XCF group during XRF takeover.

In an XRF environment, OTMA clients are equivalent to Class 3 terminals and are not automatically reconnected to IMS. The clients detect that IMS has left the XCF group and wait for IMS (or the XRF alternate) to rejoin the group. Then the clients send new client-bid requests to IMS.

The alternate system in an XRF complex does not track when resynchronization has begun or the resynchronization of a particular tpipe. If IMS fails during resynchronization, the client should detect the IMS failure, and try to resynchronize with the new XRF active system.

IMS Queue Control Facility and OTMA

The IMS Queue Control Facility (QCF) supports OTMA messages. You can use QCF to switch between all supported IMS releases, or between Shared Queues and non-Shared Queues.

You can access QCF both online with an ISPF interface and with control statements in a BMP environment.

TMEMBER and TPIPE are the operands for the INCLUDE and EXCLUDE control statements.

tmember
A 1- to 16-character OTMA target member (client) name. You can generically specify groups of names that begin with the same characters by using an asterisk (*) after the characters that are the same. An asterisk as the first character will include or exclude **all** OTMA transactions.

tpipe
A 1- to 8-character OTMA transaction pipe name. You can generically specify groups of names that begin with the same characters by using an asterisk (*) after the characters that are the same.

To selectively recover OTMA messages, use the INCLUDE and EXCLUDE control statements. The format of the INCLUDE and EXCLUDE statements with OTMA operands is:

```
INCLUDE  operand(,)

EXCLUDE  operand(,)
```

operand must start in column 10 and is one of the following:

- TMEMBER=tmember

- TPIPE=tpipe

Example: To select all OTMA messages using transaction pipe name S4A1BV6, specify:

```
INCLUDE  TMEMBER=*,
         TPIPE=S4A1BV6
```

All messages with the same tmember are grouped together, and the count is reported by the tpipe name.

Example:

```
  **** MESSAGES INSERTED BY DESTINATION ****
         BY OTMA DESTINATION
         TMEMBERNAME
           TPIPE1                              count
           TPIPE2                              count
```

If a client-bid request changes the name of the current OTMA Destination Resolution exit routine, any transactions enqueued before IMS terminates that are then reprocessed by the Message Requeuer might not use the changed exit routine name. Inserts to alternate PCBs use the exit routine name in the client descriptor.

With QCF, you can identify a category of message as well as the message type. The following table describes category parameters, and the supported message types and keywords associated with the parameter.

Table 145. Selecting messages by category type

Category parameter	Description	Supported message types and keywords
DESTYPE	Checks the destination of a message for a selected message type	APPC, LTERM, MSC, OTMA, LTRAN, RTRAN, TRANS, VSP
SRCETYPE	Checks the source of the message for a selected message type	APPC, MSC, OTMA, VSP
MSGTYPE	Checks the source or destination of the message for the selected message type	APPC, LTERM, MSC, OTMA, VSP

Related Reading: For more information about message selection by category in QCF, refer to *IMS Queue Control Facility for z/OS User's Guide*.

Using shared queues with OTMA

This topic describes general information about using IMS shared queues with OTMA.

To ensure delivery of alternate PCB processing, enable OTMA on all IMS systems; assign each IMS system in the shared queue group a unique z/OS cross-system coupling facility member name.

Use the **/DISPLAY TRANS** ALL QCNT to view all the OTMA transactions currently in the shared queue group waiting to be processed. Use **/DISPLAY QCNT** OTMA MSGAGE 0 to view all the OTMA outbound messages on the shared queues.

Note: If OTMA outbound messages remain on a shared queue when the IMS system that was processing them is shutdown normally and then cold started, the outbound messages become stranded on the shared queue and cannot be delivered. However, if a normal shutdown is followed by a warm start, any outbound messages on the shared queue at the time of the shutdown can be delivered.

For OTMA hold-queue capable clients, such as IMS Connect, using the OTMA super member function can ensure that outbound messages do not become stranded on a shared queue and can be delivered to the client.

As the result of a temporary shortage in the HIOP storage pool, you might receive message DFS1269E, which notifies you of an internal IMS failure to register a shared queue resource. To re-register the shared queue resource for OTMA, issue the IMS commands /STOP OTMA and /START OTMA.

In a shared queues environment, IMS removes idle, non-queued, non-synchronized transaction pipes after three system checkpoints.

If you send OTMA ALTPCB messages to a remote IMS system via an IMS-to-IMS TCP/IP connection from the shared-queues environment, the transaction messages can be sent to the remote IMS system only from the IMS system that is directly connected to the IMS Connect instance that is managing the TCP/IP connection.

For OTMA commit-then-send messages in a shared queues environment, a new ITASK under a new OID TCB is created to process the messages, separately from the main ITASK under OIC TCB that is used for other jobs.

Related concepts

"Using the OTMAYPRX user exit and DFSYDRU0 exit routine to determine destination" on page 782
Transaction pipe names can be the same as IMS LTERM names or APPC/IMS TP names.

Related reference

DFS1269E message information (Diagnosis)

Related information

DFS1269E (Messages and Codes)

OTMA commit-then-send messages

OTMA commit-then-send (commit mode 0 or CM0) messages can be processed on any IMS system in the shared queues group. Program-to-program switches can also be run on any IMS.

A new IMS ITASK under operator identification task control block (OID TCB) for each shared queues front-end IMS is created to boost the performance for OTMA message processing between front-end IMSs and other back-end IMSs.

OTMA unsolicited messages

OTMA clients must connect to every IMS system in the shared queue group in order to receive unsolicited messages. The OTMA client connections are necessary because transactions that might cause unsolicited messages can run on any IMS within the shared-queues group.

If the IMS that processes an unsolicited message (the back-end system) is a different IMS than the one that receives the message, the unsolicited message is delivered by the back-end system. Therefore, OTMA must also be enabled on the back-end IMS.

OTMA send-then-commit messages

OTMA send-then-commit messages can also be processed on any IMS system in the shared queue group.

Synchronous and asynchronous transactions created by a program-to-program switch from an input synchronous transaction always run on the same IMS system as the transaction that initiated the program-to-program switch.

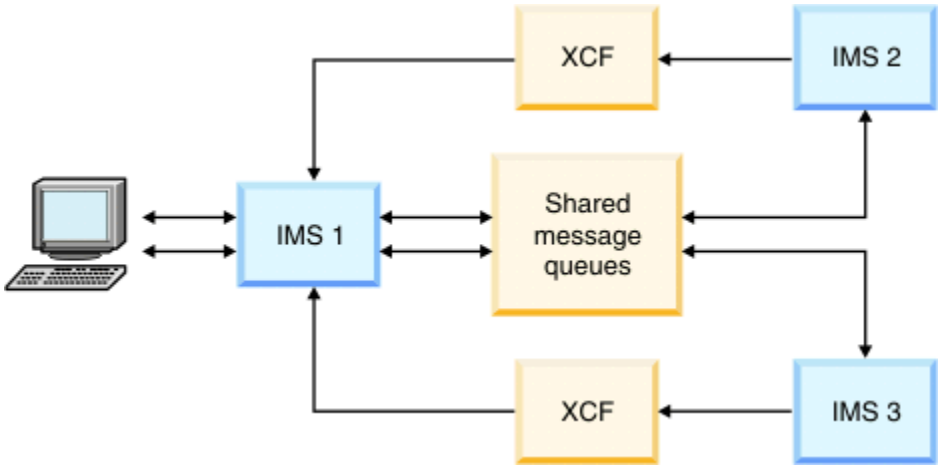


Figure 140. OTMA messages being processed on multiple IMS systems in a shared-queues group

In addition, program-to-program switching is not allowed for protected conversations (sync level 2).

Synchronous transactions which use send-then-commit processing support the following commit levels:

- NONE
- CONFIRM
- SYNCPT

Asynchronous transactions which use commit-then-send processing support the following commit levels:

- RESYNC
- NO RESYNC

The commit levels for synchronous and asynchronous transactions are shown in the following figure.

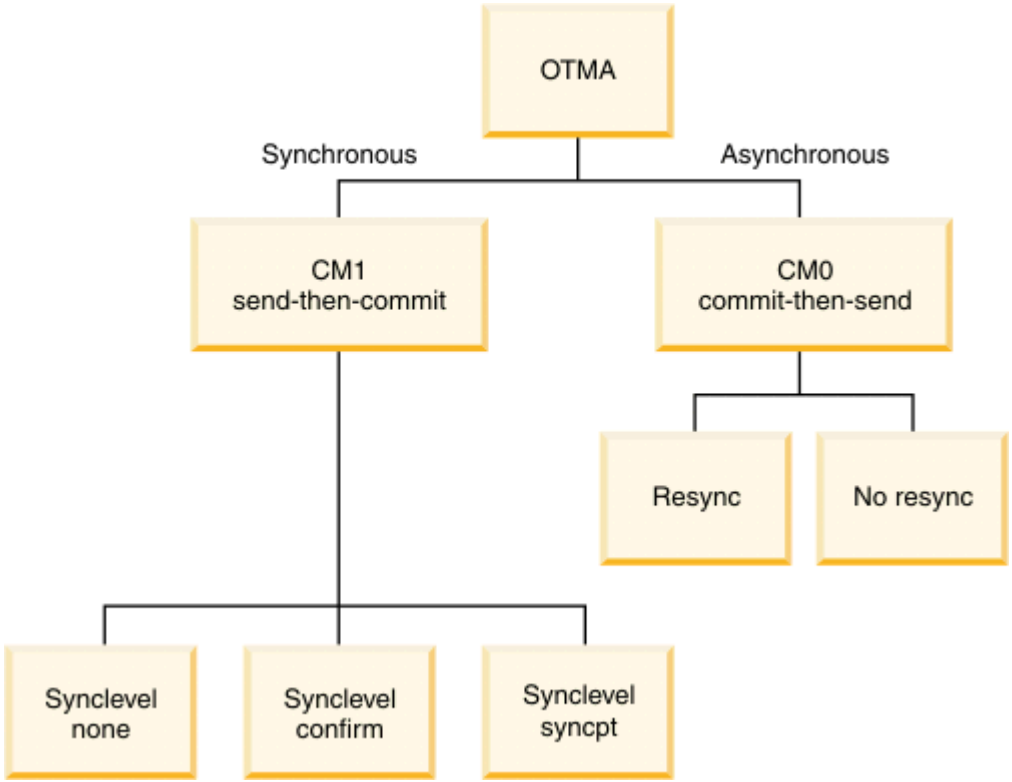


Figure 141. Synchronous and asynchronous transactions and their respective commit levels

Use the /DISPLAY ACTIVE command to determine whether the shared queue function for OTMA send-then-commit is active.

Using other IMS commands

The IMS command /DISPLAY TMEMBER *membername* TPIPE *tpipename* QCNT shows the tpipe status and the output message queue count in a shared queue for a particular IMS system.

Related concepts

"OTMA program-to-program switch processing" on page 817
Two types of message switch occur in OTMA: commit-then-send, and send-then-commit.

Retrieving ALTPCB output from a back-end IMS system

In a shared-queues environment, unless you take additional steps, the alternate PCB (ALTPCB) output placed on the shared queue by a back-end IMS system has affinity to the back-end system. This affinity prevents the originating front-end IMS system from retrieving the output unless you take additional steps.

To enable a front-end IMS system to retrieve ALTPCB output that is placed on the shared queue by a back-end IMS system, you can use the following solutions:

1. Enable the super member function and code the OTMA client to issue a resume tpipe call at the front-end IMS system.

   All retrieval options of the resume tpipe call are supported in the shared-queues environment.
2. Disable the back-end affinity. This option is useful for OTMA clients that do not use the super member function, such as IBM MQ.

   You can disable the back-end affinity by either of the following methods:

   - In the OTMA client descriptor in the DFSYDTx PROCLIB member of the front-end IMS system, specify ALTPCBE=YES.
   - In third byte of the state data prefix of the input transaction message, set the TMAMALTB flag (X'01').

   When either of these two options are specified, the ALTPCB output generated at the back-end IMS system is delivered to the front-end IMS via the shared queue.
3. Enable the SENDALTP function for IMS Connect customers. By using this solution, an ALTPCB output is sent back to the initiating IMS Connect client without the need of a resume tpipe call.

Related tasks

"Sharing asynchronous commit-then-send output: the OTMA super member function" on page 847
Hold-queue-capable OTMA clients, such as IMS Connect, can share asynchronous commit-then-send (CM0) output messages by enabling the OTMA super member function. The OTMA super member function is specifically designed to support multiple instances of IMS Connect in a z/OS Sysplex Distributor environment.

Delivering ALTPCB output without RESUME TPIPE calls for IMS Connect

When IMS Connect is identified as the destination of an IMS ALTPCB output message, IMS queues the output to an OTMA TPIPE hold queue instead of delivering it. This ALTPCB output can be kept in IMS until an IMS Connect client issues a RESUME TPIPE call to retrieve it.

When you call the SENDALTP function, IMS delivers an ALTPCB output to the originating IMS Connect client for its send-receive (CM0) call without the need of a RESUME TPIPE call. In order for the SENDALTP function to work, the ALTPCB output should be routed to the same IMS Connect client that initiates the call.

The supported message flow is: TCP/IP APP → IMS CONNECT X → IMS → IMS CONNECT X → TCP/IP APP. This SENDALTP function won't work if the ALTPCB output initiated by an IMS Connect client is routed to a different IMS Connect client or to a non-IMS Connect destination.

Activating the SENDALTP function

To activate the SENDALTP function in IMS, you can choose from using message level activation or datastore level activation.

For message level activation, you can use one of the following methods:

- Set the IMS Connect IRM flag, IRM_F2_SNDALTP, for a user-written IMS Connect client
- Specify SENDALTP for OTMA destination descriptor in DFSYDTx PROCLIB member
- Set the SENDALTP output flag for IMS DFSYDRU0 user exit

For datastore level activation, you can use one of the following methods:

- Issue CREATE IMSCON or UPDATE IMSCON commands to turn on the SENDALTP function for data stores
- Specify SENDALTP to activate the function in IMS Connect configuration file for data stores
- Specify SENDALTP to activate the function using OTMA client descriptor

The order of precedence of activating the SENDALTP function is as follows:

1. Message level activation from an IMS Connect client
2. Message level activation through IMS OTMA destination descriptor or IMS DFSYDRU0 user exit
3. Datastore level activation through IMS Connect configuration or IMSCON type 2 commands
4. Datastore level activation through IMS OTMA client descriptor

Handling SENDALTP with OTMA destination descriptors

When OTMA destination descriptor is used to handle the ALTPCB output message, the SENDALTP function will be disabled in the following cases:

- The TYPE= parameter is not IMSCON.
- The TMEMBER= parameter of the OTMA destination descriptor is not the initiating IMS Connect member.

Specifying a super member doesn't affect the SENDALTP function. For example, when both SMEM=YES and SENDALTP=YES are specified, IMS sends the ALTPCB output back to the initiating IMS Connect client instead of queuing it to the super member. When the SENDALTP function is activated for an ALTPCB output for IMS Connect, the ALTPCB output TPIPE name will be the input TPIPE name of the initiating commit-then-send send-receive call, instead of the TPIPE name specified in the destination descriptor.

Handling SENDALTP with OTMA DFSYDRU0 user exit

When OTMA DFSYDRU0 user exit is used to handle the ALTPCB output message, the SENDALTP function will be disabled if the exit sets a different output destination member or datastore from the initiating IMS Connect client.

Once the SENDALTP function is activated, if you specified the TPIPE override name in the output parameter of the exit, it will be omitted. The input IMS Connect TPIPE name will be used to send back the ALTPCB output instead.

Restrictions on using the SENDALTP function

Using the SENDALTP function has the following restrictions:

- SENDALTP can only be enabled for IMS Connect. Other OTMA clients, such as MQSeries, cannot activate this function.
- SENDALTP only supports commit-then-send send-receive calls from IMS Connect. Other input messages, such as send-then-commit input messages and commit-then-send send-only calls, cannot enable the SENDALTP function.
- SENDALTP function cannot be used to process an IMS ALTPCB output that is initiated by non-IMS Connect clients.
- If an OTMA destination descriptor or an IMS DFSYDRU0 user exit routes the IMS ALTPCB output to a non-OTMA destination or a different OTMA client, the SENDALTP function will be disabled.

Attention: Don't activate the SENDALTP function for IMS Connect if a commit-then-send send-receive call generates both I/O PCB outputs and ALTPCB outputs. If you do so, IMS will deliver both types of output messages to IMS Connect. IMS Connect will only accept the first arrived output message and reject the subsequent output messages. The rejected output messages will then be queued to an OTMA TPIPE hold queue or be purged if asked.

IMS termination and OTMA

When IMS terminates, OTMA attempts to notify OTMA clients and might take additional actions depending on the circumstances.

Related tasks

Shutting down IMS (Operations and Automation)

OTMA client notification of IMS termination

OTMA clients are notified when IMS terminates to prevent them from sending new requests after IMS shuts down.

During a planned IMS shutdown, OTMA sends the protocol commands TMAMMNTR (X'3C' Resource Monitor) and TMAMCSPA (X'14' Suspend TPIPE) to all OTMA clients.

During an unplanned IMS shutdown, OTMA leaves the XCF group during abend processing and XCF exit routines notify the OTMA clients.

Related reference

“Server state protocol command” on page 878
The state data for the server state protocol command is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

“Message-control information section” on page 859
For every OTMA message, you must provide message-control information in the first section of the OTMA message prefix.

IMS termination and IMS-to-IMS TCP/IP messages

If a tpipe is in a WAIT_R state when IMS terminates normally or abnormally, OTMA reroutes the messages on the tpipe to the default timeout queue, DFS$$TOQ, during restart processing.

The WAIT_R status of a tpipe indicates that OTMA is waiting for an acknowledgment of receipt for a message sent to a remote IMS system.

You can clear the WAIT_R status of a tpipe by issuing the **/STOP TMEMBER** TPIPE command.

You can use the /DISPLAY TMEMBER command to determine which OTMA tpipes have a WAIT_R status.

Related reference

/DISPLAY TMEMBER command (Commands)

OTMA restrictions and requirements

A number of general restrictions and requirements apply to OTMA.

Restrictions

The maximum total length of all prefixes for an OTMA message is 4096 bytes. This length does not include any application data.

Existing IMS application programs that use SET0 calls might not run as expected. APPC/IMS application programs using SET0 calls might require modification to use implicit OTMA support.

OTMA does not support the IMS Message Format Service (MFS). However, the MFS message output descriptor (MOD) name can be specified by the client in the prefix of an OTMA message.

OTMA does not support IMS Front-End Switch.

OTMA messages cannot be encrypted.

OTMA has read only access to the main storage data base (MSDB). No update access is available to MSDB from OTMA.

OTMA does not operate in the IMS DBCTL environment.

OTMA does not allow IMS terminal control commands, such as /FORMAT, /HOLD, /RCL, and /SIGN commands.

Requirements

IMS conversational and Fast Path transactions must be defined as send-then-commit. Existing Fast Path applications can run with OTMA.

A transaction from an IMS terminal (for example, a SLU 2 terminal) cannot route output directly to a client, but must use an OTMA Destination Resolution user exit (OTMAYPRX).

All user IDs must be verified by RACF , unless the client specifies no security checking in the security-data section of the message prefix.

IMS modules that contain z/OS cross-system coupling facility (XCF) macros must be reassembled for new releases of IMS.

OTMA protected conversation messages that are part of the same z/OS Resource Recovery Services (RRS) unit of recovery and that access common database resources must be scheduled and executed on the same IMS system.

IMS shares locks and database buffers between protected conversation transactions, but this support is restricted to within a single IMS system. It does not function across shared queues. You must ensure such messages schedule on the same IMS system.

Managing system resources and OTMA

OTMA uses IMS system resources. A variety of factors affect how well OTMA uses those resources and there are a number of ways to monitor that usage.

The following topics describe these system resource considerations.

Administering OTMA tmembers

IMS provides commands to get information about and dynamically modify OTMA target members (tmembers).

Use the **START TMEMBER** and **STOP TMEMBER** commands to start and shut down specific OTMA tmembers and tpipes. The commands also provide parameters to change the size of the input message hold queue and timeout length for enqueued CM1 and CM0 messages.

The **/STOP TMEMBER** command also clears the wait state (WAIT_A, WAIT_H, or WAIT_S) of all CM0 messages on the tpipe that are waiting for either an ACK or a NAK message from the client.

You can retrieve detailed information about some or all tmembers with the **DISPLAY TMEMBER** command. You can specify **DISPLAY TMEMBER ALL** to retrieve a report for all of the configured OTMA tmembers and their associated tpipes, or qualify the command with specific tmember and tpipe names to narrow the search. The response output shows the processing mode for each tpipe, security level, status, workload statistics, and other information.

The enqueue and dequeue counts on a tpipe are updated only for CM0 output messages. The counts are not updated for CM1 messages regardless of the sync level because CM1 messages are not recoverable.

You can also retrieve a list of OTMA tmembers with stopped tpipes by using the **DISPLAY STATUS TMEMBER** command. This command is useful for determining whether any tmembers require

administrator attention. The command also retrieves the name of the super-member for each tmember, if one is configured.

You might need to manually dequeue stalled or stale messages from an OTMA tpipe. For example, you might want to purge stale messages from a tpipe after modifying a client application so that new messages can be retrieved instead. You can use the **DEQUEUE** command to purge messages from a specific tpipe. The command provides options to dequeue either the first enqueued message for the tpipe or all enqueued messages.

If you need additional information about a tmember or a specific tpipe, you can enable the OTMA client activity trace with the **TRACE TMEMBER** command. You can also turn on tracing for an entire super member. Use the **DISPLAY TRACE TMEMBER** command to determine which tmembers and tpipes are currently being traced for an OTMA client. The tracing information is available in the OTMA log records.

A temporary tpipe is created when you issue a **/TRACE** *tpipe* or **/STOP** *tpipe* command against a tpipe that does not exist. A temporary tpipe is converted to a permanent tpipe if an input message reaches IMS through the tpipe or if an output message is queued to the output queue of the tpipe.

When you issue a **/DISPLAY** *tpipe* command against a temporary tpipe, the status of TMP is displayed.

Related reference

/DISPLAY TMEMBER command (Commands)
/DISPLAY STATUS command (Commands)
/DEQUEUE command (Commands)
/START TMEM command (Commands)
/STOP TMEM command (Commands)

Buffer pool usage for OTMA

If an IMS-OTMA environment has heavy OTMA traffic, a significant increase in LUMP and HIOP pool usage can occur.

Because LUMP and HIOP pools are allocated from private storage, you might need to increase the size of the IMS control region. Also, certain OTMA control blocks are allocated from extended common service area (ECSA), another limited resource.

Recommendation: Increase the ECSA size according to your workload. For example, if a client is sending more than 20 messages over 100 tpipes within a few seconds, try increasing the IMS control region size to 200 MB or more, and increase the ECSA size to 50 MB or more. If you cannot increase the IMS control region size or the ECSA size, try balancing your workload to allow IMS to reuse its buffers more effectively.

Collecting OTMA checkpoint statistics

You can collect OTMA checkpoint statistics with the X'451A' record and the X'451B' record.

The two X'45' statistic record types are written with the X’45’ records during checkpoint processing. The X'451A' record is written for OTMA global information and the X'451B' record is written for member information for each OTMA member. These records are mapped by DSECTs in macros DFSL451A and DFSL451B. These statistics include TPIPE and YTIB information such as current number in use, high use number since last checkpoint, and upper limits. The current number of YQABs in use is included in the X'451A' record. This YQAB value is the total YQAB count for the primary YQAB of TPIPEs, hold queue YQAB of TPIPEs, and MQAB for MULTIRTP function.

Related concepts

OTMA checkpoint statistics enhancement (Release Planning)

Dependent region occupancy and OTMA

A send-then-commit transaction remains in a dependent region while the output is being sent (before a sync point occurs). You can get additional information about a specific dependent region that is in an OTMA-related wait state by using the **DISPLAY ACT** command.

The following states (from the STATUS response field) indicate that an OTMA-related activity is in progress:

- WAIT-CALLOUT
- WAIT-RRS PC
- WAIT-SYNCPOINT
- TERM-WAIT SYNCPT
- TERM-WAIT RRS

If the region is waiting for a synchronous program switch response, the amount of time remaining before the region times out the request is displayed in the END TIME field of the output from the **DISPLAY ACT REGION** command.

Recommendations:

- If many of your transactions are send-then-commit (CM1) transactions, increase the number of dependent regions to improve throughput performance.
- For the CM1 messages that use synclevel=confirm or synclevel=syncpt, specify a timeout interval to free dependent regions if an OTMA client does not provide the required ACK or NAK response within a reasonable amount of time.
- Use as many commit-then-send (CM0) OTMA transactions as possible.

Related tasks

"Specifying acknowledgment timeout intervals for OTMA messages" on page 801
You can specify an ACK timeout interval that determines how long OTMA waits for an ACK or NAK acknowledgment for OTMA output messages.

Related reference

/DISPLAY ACT command (Commands)

Displaying the current transaction workload

You can view information about the number and type of OTMA messages that are currently being processed by IMS by issuing the type-2 IMS Command **QUERY OTMATI**.

Issuing the **QUERY OTMATI** command without specifying any parameters shows the current number of OTMA messages on each IMS system in an IMSplex listed with the OTMA client and the tpipe used to submit the messages. For example, in an IMSplex with three IMS systems the following information could be returned by **QUERY OTMATI**:

```
TMember          TPipe    MbrName    CC MsgCnt
MQ               APPLA    IMSA        0    102
MQ               APPLB    IMSB        0    201
WAS              APPLC    IMSB        0    301
```

By specifying the parameters of **QUERY OTMATI**, you can filter and refine the information returned by the command.

For example, if you need to know if messages are waiting too long to be processed by IMS, the MSGAGE(*nnn*) subparameter returns a count of all messages in an IMS system that are older than *nnn*, where *nn* is a clock time specified in seconds. In the following example, **QUERY OTMATI MSGAGE(3)** was specified and the message count for each IMS system includes only those messages that are older than three seconds:

```
TMember          TPipe    MbrName    CC MsgCnt MsgAge
MQ               APPLA    IMSA        0     50      3
```

```
MQ              APPLB    IMSB       0     80      5
WAS             APPLC    IMSB       0     100     4
```

In the example above, the number shown in the MsgAge column is the age of the most recently submitted message among all messages that are older than three seconds. For example, among the 80 messages submitted by MQ on tpipe APPLB to IMSB that are older than three seconds, the message that has been processing for the least amount of time has been processing for five seconds.

The **QUERY OTMATI** command includes over 25 parameters and subparameters that return information about the various components and resources that are associated with the processing of messages from OTMA clients. In addition to message counts and ages, you can also retrieve information about such things as:

- The transaction or application program scheduled to process the message
- Security information
- Context tokens for two-phase commit processing
- OTMA commit mode
- Synchronization level
- Correlator tokens for callout processing
- Timeout setting for acknowledgments from the client
- The conversation ID of messages that belong to a conversational transaction

Impact of OTMA message TIBs on storage

For each message received from an OTMA client, IMS creates a *transaction instance block* (TIB).

Each TIB requires approximately 8 KB of extended private area storage.

You can view information about the TIBs and the OTMA messages that are currently being processed by IMS by issuing the type-2 IMS Command **QUERY OTMATI**. The information you can view includes message counts, message ages, and much more.

Normally TIBs are freed and the storage released by IMS either after IMS enqueues the commit-then-send (CM0) input message of TIBs created for commit-then-send messages or after IMS returns a send-then-commit (CM1) output message of TIBs created for send-then-commit messages.

An excessive number of TIBs is usually caused by either a message flood condition or orphan TIBs.

Message flood conditions usually occur as a result of:

- A stopped program
- The unavailability of message processing regions
- The rate of incoming messages exceeding the rate at which IMS can process the messages

To help avoid a message flood condition, you can enable message flood detection by specifying a maximum number of OTMA messages that IMS can process at any one time. IMS determines the number of messages that are currently being processed by counting TIBs. When message detection is enabled, IMS issues messages when the number of TIBs nears the maximum.

Orphan TIBs

There are circumstances in which IMS cannot free a TIB. When this happens, the TIB persists in storage as an *orphan TIB*.

Orphan TIBs are created when IMS processes an OTMA transaction, but does not generate either a send-then-commit output message or a DFS2082 message in place of the send-then-commit output message. For example, if a send-then-commit message triggers a program-to-program switch to an asynchronous transaction that is defined as a non-response mode transaction, and if OTMAASY=Y, IMS cannot delete the TIB even thought the switched-to transaction generates a response to the IOPCB.

IMS includes orphan TIBs in its count of total TIBs when message flood is enabled. If OTMA input has been stopped due to a message flood condition, you can issue the **/DISPLAY OTMA** or **/DISPLAY TMEMBER** command to see the number of TIBs. If there is still a high number of TIBs after the OTMA input has been stopped, it is possible that there are orphan TIBs.

You can remove orphan TIBs related to IMS conversational transactions. You can identify orphan TIBs that are related to conversational transactions by issuing the command **/DISPLAY CONVERSATION**. If an orphaned TIB exists for the conversation, you can then issue the command **/EXIT CONVERSATION** to remove the orphaned TIB.

You can also remove orphan TIBs related to conversational transactions in OTMA by specifying the **ENDCONV=** parameter of the DFSOTMA descriptor in the DFSYDT*x* member of the IMS PROCLIB data set. If the conversational transaction remains idle for the specified period of time after the prior iteration of the conversational transaction completes, OTMA ends the transaction and releases IMS resources that were allocated for the transaction.

Related concepts

"Displaying the current transaction workload" on page 794
You can view information about the number and type of OTMA messages that are currently being processed by IMS by issuing the type-2 IMS Command **QUERY OTMATI**.

"Message flood detection" on page 796
If IMS cannot process the messages received from an OTMA client quickly enough or at all, the build up of messages in the IMS system can result in a *message flood* condition. When a message flood condition occurs, the storage required by both the messages and the transaction instance blocks (TIBs) that IMS uses to process the messages, can exhaust the below-the-line storage and potentially cause a z/OS S40D system abend.

Related reference

DFSOTMA descriptor syntax and parameters (System Definition)

IMS message queue data set size and OTMA

Messages entering IMS from OTMA contain both the OTMA message prefix and other existing IMS message prefixes.

The OTMA message prefix is variable in length. Excluding the user data section, the OTMA message prefix can become very large, sometimes over 200 bytes in length. The OTMA message prefix, including the user data section, is stored on IMS message queue data sets, which increases usage of the queue buffer pool.

Tip: Because of this increase in queue buffer pool usage, try to increase the size of the message queue data sets.

If the security-data section of the OTMA message prefix contains network security credentials, the size of the OTMA message can increase by up to 504 bytes. Therefore, consider increasing the size of the SHMSG and LGMSG message queue data sets and the size of the message queue pool.

Message flood detection

If IMS cannot process the messages received from an OTMA client quickly enough or at all, the build up of messages in the IMS system can result in a *message flood* condition. When a message flood condition occurs, the storage required by both the messages and the transaction instance blocks (TIBs) that IMS uses to process the messages, can exhaust the below-the-line storage and potentially cause a z/OS S40D system abend.

To avoid the problems associated with a message flood condition, OTMA can monitor the number of input messages that are waiting to be processed in the IMS system. When monitoring is active, OTMA monitors both the number of input messages from each OTMA client and the total number of input messages from all OTMA clients combined. When the number of input messages from an individual OTMA client reaches its defined maximum, OTMA suppresses new input messages from that client. When the total number of input messages reaches the global maximum, OTMA issues a warning message, but still accepts new input messages.

You can modify the maximum number of input messages for both individual OTMA clients and for all OTMA clients combined. You can specify from 200–9999 as the maximum number of allowable messages or you can disable message flood detection by specifying 0.

The default maximum number for OTMA clients is 5000. The default maximum number for all OTMA clients combined is 10,000.

When the number of messages in the IMS system reaches 80% of the maximum number defined for either an OTMA client or for all OTMA clients, OTMA issues a server state protocol command to the appropriate OTMA clients and IMS issues message DFS1988W to the master console. IMS also issues DFS1988W at every 5% increase thereafter until the maximum number is reached.

When the number of messages in the IMS system for an individual client exceeds the maximum, OTMA issues another server state protocol command and IMS issues message DFS1989E. OTMA then rejects all subsequent input messages from the OTMA client until the message flood condition is resolved.

When the number of messages in the IMS system falls to 50 percent of the maximum, OTMA issues a new server state protocol command to the OTMA clients notifying them that the flood condition has been relieved.

To activate global flood control to suppress new OTMA input transactions, complete one of the following tasks:

- Specify the global flood limit value **INPT=** for the DFSOTMA member in the OTMA client descriptor for all the OTMA members.
- Issue the IMS command **/START TMEMBER ALL INPUT** *nmbr*.

Values specified in the OTMA client descriptor can be overridden by the **/START TMEMBER** command. If the client-bid request specifies a smaller number, the client-bid request can also override the value specified in the OTMA client descriptor.

Related concepts

"Impact of OTMA message TIBs on storage" on page 795
For each message received from an OTMA client, IMS creates a *transaction instance block* (TIB).

Monitoring tpipe usage

Transaction pipes (tpipes) use significant amounts of IMS resources and processing time, so try to limit the number of tpipes that are created for each tmember.

IMS removes transaction pipes after they have been idle for three consecutive system checkpoints, except in the following circumstances:

- Commit-then-send messages are queued on the tpipe or the tpipe hold queue.
- The tpipe is stopped.
- A trace is set on the tpipe.
- The tpipe is a synchronized tpipe, such as a tpipe used by MQSeries for commit-then-send input transactions.
- The tpipe is in a WAIT state for a resume tpipe request that specified either the AUTO or the SINGLE-WAIT options.
- The tpipe is in an MCP state, which indicates that the tpipe is running in a shared queues environment and might have output messages on the global queue.

  Tip: If no messages are queued to the TPIPE but the MCP status is displayed for the TPIPE so that the tpipe cannot be removed, issue the /DISPLAY TMEMBER *tmembername* TPIPE *tpipename* QCNT command or the /DISPLAY TMEMBER *tmembername* QCNT command to reset the MCP status.
- The tpipe is being scanned by IMS.

You can use the **/DISPLAY TMEMBER TPIPE** command to see whether a tpipe cannot be removed by IMS because one of the circumstances in the preceding list is true for the tpipe.

One way you can control the number of tpipes that are created for a particular OTMA client, is to set a maximum allowable number of tpipes for each OTMA client and for the IMS system.

Tpipe limits for individual OTMA clients

A maximum number of tpipes is specified for an OTMA client by specifying the MAXTP parameter on the OTMA client descriptor in the DFSYDTx member of the IMS.PROCLIB data set.

When the MAXTP parameter is specified for an OTMA client, OTMA monitors the number of tpipes created for the client and issues warnings when the number reaches certain levels.

After a MAXTP value is set for an OTMA client, OTMA monitors the number of tpipes that are created for the client. If the total number of tpipes reaches 80% or a user-specific percentage through the **MAXTPWN** parameter of the maximum allowable number, IMS issues the warning message DFS4382W to the system console and the MTO. IMS also issues a protocol command **TMAMMNTR** (X'3C' Resource Monitor) message with the warning status of the server to the OTMA client.

If the number of tpipes reaches 100% of the maximum allowable number, IMS issues error message DFS4383E to the system console and the MTO. Any input transactions that require a new tpipe are rejected with a NAK with OTMA sense code X'29'.

After the warning or error messages is issued and the total number of the tpipes for the client drops 50% or a user-specified percentage through the **MAXTPRL** parameter of the maximum allowable number, IMS issues message DFS4384I to indicate that the number of tpipes returned to normal.

Tpipe limits in the IMS system

You can define a global TPIPE warning threshold for all of the OTMA clients by defining an OTMA client descriptor with the name DFSOTMA and specifying the MAXTP parameter. The DFSOTMA descriptor is the OTMA system client descriptor that defines values for the IMS system that apply to all OTMA clients.

When the number of tpipes in the IMS system reaches 80% or a user-specified percentage of the DFSOTMA MAXTP value through the DFSOTMA **MAXTPWN** parameter, IMS issues message DFS4515W to both the system console and MTO. OTMA also issues a protocol message to all OTMA clients. After that, OTMA continues creating new tpipes until the number of tpipes reaches the maximum number as defined by the DFSOTMA **MAXTP** value.

If the total number of tpipes in the IMS system reaches 100% of the DFSOTMA MAXTP value, OTMA rejects all new tpipe creation requests from any OTMA members with a NAK that contains the sense code X'29'. IMS sends DFS4516E error message to the system console and the MTO, and notifies all OTMA clients with an OTMA protocol message.

After a DFS4515W or DFS4516E is issued and the total number of monitored tpipes in the system drops down to 50% or another user-specified level of the global tpipe warning threshold through the DFSOTMA **MAXTPRL** parameter, IMS issues message DFS4517I to indicate that the number of tpipes returned to normal.

If the OTMA DFSOTMA system client descriptor does not define a global maximum for the number of tpipes in an IMS system, and one or more OTMA clients have a maximum allowable number of tpipes defined in their OTMA client descriptors, the highest MAXTP value among all of the client descriptors serves as a global warning threshold for the IMS system. When the total number of tpipes in use by all OTMA clients that are subject to tpipe monitoring reaches the global warning threshold, IMS issues warning message DFS4385W to the system console and the MTO.

After a global warning is issued and the total number of tpipes drops down to 80% of the global warning threshold, IMS issues message DFS4386I to indicate that the number of tpipes returned to normal.

X'3C' protocol command notifications

OTMA also sends out the X'3C' protocol command to the OTMA clients at the various warning, error, and relief thresholds. Upon receiving the X'3C' protocol command for a warning or error, the client applications can reroute any subsequent transactions to a different IMS system as appropriate. When the

number of tpipes drops below the relief threshold, the X'3C' protocol command is issued again with the warning or error flag turned off.

Displaying information about the number of tpipes

The IMS commands **/DISPLAY OTMA** and **/DISPLAY TMEMBER** can show the current number of tpipes for the OTMA clients that have a MAXTP value set. If an OTMA client reaches the maximum allowable number of tpipes, the command output shows MAX TPIPE as the USER_STATUS for the OTMA client.

The global warning threshold set by the highest MAXTP value among multiple OTMA clients is displayed under the output field TPNCT for the IMS server.

Reducing tpipe storage by using the lightweight tpipe function

You can enable the lightweight tpipe function by specifying **LITETP=**YES in the DFSOTMA descriptor in the DFSYDTx IMS.PROCLIB member. When this function is enabled, less storage is used when a tpipe is created in a shared queues back-end IMS system to process front-end input transactions. Specifying **LITETP=**YES enables IMS to support more tpipes. If **LITETP=**YES is specified, message DFS7411I is issued on IMS initialization to indicate that this function is enabled.

Because a lightweight tpipe requires less storage than a regular tpipe, a weighting factor is used on back-end tpipes when calculating the tpipe count for tpipe flood control. The weighting factor is the percentage of the lightweight tpipe storage size relative to the regular tpipe storage size, which is usually 28%. See the description of the **LITETP=** parameter in DFSOTMA descriptor syntax and parameters for the calculation of the adjusted tpipe count.

Related reference

/DISPLAY TMEMBER command (Commands)

DFSOTMA descriptor syntax and parameters (System Definition)

Monitoring system resources with OTMA

OTMA monitors IMS system resources that are used to process OTMA transactions and notifies OTMA clients about how well the IMS system is processing OTMA transactions.

If an OTMA client receives a notification that the IMS system is not processing OTMA message normally, the OTMA client can then take appropriate action, such as rerouting OTMA transaction messages to a different IMS system.

OTMA sends notifications to OTMA clients as a *server state protocol command*. OTMA issues the server state protocol command in the following circumstances:

- When an OTMA client establishes a tpipe connection
- When a significant change occurs in the ability of IMS to process OTMA messages
- As a *heartbeat message* at 60 second intervals

The server state protocol message categorizes the overall state of the IMS system by using the following states.

Normal state (X'03')
IMS is available and is processing OTMA messages normally.

Degraded state (X'02)
IMS is processing OTMA messages slowly. OTMA issues a degraded state protocol command when one or more conditions indicate that IMS is not processing OTMA messages as quickly as it should.

Unavailable state (X'01)
IMS can no longer accept OTMA transactions for processing. OTMA issues the unavailable state protocol command to alert the OTMA client that one or more severe conditions prevent IMS from processing OTMA messages.

In addition to notifying the client of the overall state of IMS processing, if the IMS processing is in either a degraded or unavailable state, the server state protocol command can include additional information about the condition of specific resources associated with the degraded or unavailable state.

The server state protocol command is identified by a value of X'3C' in the protocol command type field (TMAMCTYP) in the message control information section of the OTMA message header. The server state protocol command itself is delivered in the state data section of the OTMA header and mapped with the TMAMRSIM field prefix in the DFSYMSG macro.

You can view information about OTMA clients, OTMA tpipe connections, and the OTMA messages that are currently being processed by IMS by issuing the type-2 IMS Command **QUERY OTMATI**.

Related concepts

"Displaying the current transaction workload" on page 794
You can view information about the number and type of OTMA messages that are currently being processed by IMS by issuing the type-2 IMS Command **QUERY OTMATI**.

Related reference

"Server state protocol command" on page 878
The state data for the server state protocol command is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

"Message-control information section" on page 859
For every OTMA message, you must provide message-control information in the first section of the OTMA message prefix.

OTMA ACEE flood control

The OTMA accessor environment element (ACEE) flood control function prevents virtual storage in the IMS control region from running out.

When OTMA security is enabled, for performance purposes OTMA caches in memory RACF user profiles as a RACF accessor environment element (ACEE). The ACEE is then used in subsequent calls to RACF to determine the user ID's authorization to the IMS command or IMS transaction requested in the input message. Using the ACEE reduces I/O to RACF. When you enable OTMA ACEE flood control, you limit the number of RACF user profiles that can cache ACEEs, which in turn limits the number of ACEEs that are cached by OTMA.

To check whether the OTMA ACEE flood control function is enabled and to check the total number of OTMA ACEEs that are cached, issue the **/DISPLAY OTMA** command.

1. To enable OTMA ACEE flood control, in the DFSOTMA descriptor of the DFSYDTx member of the IMS PROCLIB data set, specify **TOACEE=**YES.
2. To define the maximum number of RACF user IDs that can have cached ACEEs, specify a value for the **ACEEUSR=** parameter of the DFSOTMA descriptor.

Related reference

DFSOTMA descriptor syntax and parameters (System Definition)

Removing idle tpipes

IMS scans transaction pipes (tpipes) during system checkpoint processing to determine if any tpipes can be deleted.

IMS removes transaction pipes after they have been idle for three consecutive system checkpoints, except in the following circumstances:

- Commit-then-send messages are queued on the tpipe or the tpipe hold queue.
- The tpipe is stopped.
- A trace is set on the tpipe.
- The tpipe is a synchronized tpipe, such as a tpipe used by MQSeries for commit-then-send input transactions.

- The tpipe is in a WAIT state for a resume tpipe request that specified either the AUTO or the SINGLE-WAIT options.
- The tpipe is in an MCP state, which indicates that the tpipe is running in a shared queues environment and might have output messages on the global queue.

  Tip: If no messages are queued to the TPIPE but the MCP status is displayed for the TPIPE so that the tpipe cannot be removed, issue the /DISPLAY TMEMBER *tmembername* TPIPE *tpipename* QCNT command or the /DISPLAY TMEMBER *tmembername* QCNT command to reset the MCP status.
- The tpipe is being scanned by IMS.

You can use the **/DISPLAY TMEMBER TPIPE** command to see whether a tpipe cannot be removed by IMS because one of the circumstances in the preceding list is true for the tpipe.

If IMS cannot remove a tpipe that has been idle across three consecutive checkpoints because one of the circumstances in the preceding list is true, IMS attempts to remove the idle tpipe again at the following system checkpoint.

You can view information about OTMA clients, OTMA tpipe connections, and the OTMA messages that are currently being processed by IMS by issuing the type-2 IMS command **QUERY OTMATI**.

Related reference

/DISPLAY TMEMBER command (Commands)

QUERY OTMATI command (Commands)

Specifying acknowledgment timeout intervals for OTMA messages

You can specify an ACK timeout interval that determines how long OTMA waits for an ACK or NAK acknowledgment for OTMA output messages.

ACK timeout intervals can be specified for the following types of OTMA output messages:

- Transaction messages that are sent to a remote IMS system for processing
- Some send-then-commit (CM1) response messages
- Commit-then-send (CM0) response messages

For transaction messages that are sent to a remote IMS system, if the ACK timeout interval expires, OTMA reroutes the transaction message to the timeout queue. If OTMA receives an ACK response from the local IMS Connect after a transaction message has timed out, OTMA issues a NAK with X'2B' sense code to the local IMS Connect. IMS connect discards the NAK message.

For a send-then-commit transaction that uses either synclevel=confirm or synclevel=syncpt, if OTMA does not receive the expected ACK or NAK response from the OTMA client before the timeout interval expires, OTMA aborts the transaction and IMS backs out the transaction, issues an OTMA CM1 deallocation message to the OTMA client, and issues message DFS0809E to the z/OS system console.

For a commit-then-send transaction, the action OTMA takes if an expected ACK or NAK response is not received before the timeout interval expires depends on whether the client is a hold-queue capable client, such as IMS Connect, or a non-hold-queue capable client, such as IBM MQ. For hold-queue capable clients, the action OTMA takes also depends on whether the output is the I/O PCB queue or the hold queue of a tpipe.

For a hold-queue capable client, if the output for a commit-then-send transaction is on the I/O PCB queue and the timeout interval expires, OTMA takes the following action:

1. OTMA attempts to deliver the CM0 output to a reroute tpipe.
2. If no reroute tpipe has been specified, OTMA attempts to deliver the output to a timeout queue.
3. If no timeout queue has been specified, OTMA delivers the CM0 output to the default timeout queue DFS$$TOQ.
4. Issues DFS3494E to the z/OS system console.

For a hold-queue capable client, if the output for a commit-then-send transaction is on the tpipe hold queue and the timeout interval expires, OTMA takes the following action:

1. OTMA attempts to deliver the output to a timeout queue.
2. If no timeout queue has been specified, OTMA delivers the CM0 output to the default timeout queue DFS$$TOQ.
3. Issues DFS3494E to the z/OS system console.

For non-hold-queue capable clients, OTMA takes the same action as for hold-queue capable clients when the CM0 output is on the I/O PCB queue.

For a synchronous callout requests from IMS application programs that issue the DL/I ICAL call, OTMA processes the request as a CM0 output message. OTMA sets the CM0 flag in the state data prefix and waits for an acknowledgment of receipt of the synchronous callout message. If an acknowledgment is not returned within the ACK timeout interval, OTMA takes the following action:

1. Issues return code X'100' and reason code of X'104' to the IMS application program.
2. Discards the synchronous callout request message. The message is not rerouted.
3. Issues DFS3494E to the z/OS system console.

You can specify a timeout interval for an OTMA client in the OTMA client descriptor. You can override the timeout interval in an OTMA client descriptor by specifying a different interval in the TIMEOUT parameter of the **/START TMEMBER** command. You can also override the timeout interval in the OTMA client descriptor by specifying a smaller timeout interval in the OTMA client's client bid request.

For individual transactions, you can only specify a timeout interval that is shorter than the timeout interval specified for the OTMA client.

If you do not specify a send-then-commit timeout value, OTMA uses a default value of 120 seconds.

To view the current timeout interval set for an OTMA client, issue the **/DISPLAY TMEMBER** command.

Specifying an acknowledgment timeout interval at the OTMA client level

At the OTMA client level, you can specify an ACK timeout interval in one or more places.

You can specify a timeout interval in the following places:

- The T/O= parameter of the OTMA client descriptor
- The TIMEOUT parameter of the /START TMEMBER command
- The 1–byte field at offset 65 (X'41') of the state data in the client-bid request from the OTMA client

For commit-then-send transactions, you can specify the name of a timeout queue by:

- Specifying X'04' at byte offset 45 (X'2D') in the client-bid request
- Specifying the 8-byte character name of the timeout queue at byte offset 66 (X'42').

If no timeout queue name is specified, CM0 output that times out is routed to the default timeout queue DFS$$TOQ.

When transaction messages that are destined for a remote IMS system time out, OTMA reroutes them to the DFS$$TOQ.

Related concepts

“OTMA client descriptors” on page 773
Use OTMA client descriptors to provide information about a specific OTMA client to IMS. OTMA client descriptor entries are identified in the DFSYDTx PROCLIB member by an M in column one of the descriptor entry.

Related reference

“Server-Available and Client-Bid commands” on page 868
The state data for the Server-Available and Client-Bid commands section of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

/START TMEM command (Commands)

Specifying an acknowledgement timeout on CM1 transaction messages

In the OTMA header of an input message for a send-then-commit (CM1) transaction that uses either synclevel=confirm or synclevel=syncpt, you can specify a timeout interval to limit how long IMS waits for an acknowledgement (ACK or NAK) to the output response.

To set the time out interval:

- Specify the interval in minutes using decimal integers in the 1–byte field at offset X'1E' of the message control information prefix of the transaction message.
- Set flag X'08' at byte 5 of the state data

Specifying an expiration time for transactions to OTMA

You can specify an expiration time for a transaction to reduce processing costs by preventing IMS from processing transactions that the client can no longer use.

When a transaction specifies an expiration time, OTMA monitors the transaction and, if the transaction is not processed or enqueued before the time expires, OTMA discards the transaction.

OTMA stops monitoring a transaction for expiration when an IMS application program in an IMS dependent region retrieves the transaction for either MSC, Fast Path, or conversational processing. After either of these events, the expiration time no longer applies and IMS processes the transaction and returns the output to OTMA and the OTMA client.

You can enable transaction expiration either when you define the transaction to IMS or in the OTMA message header when the transaction is submitted to OTMA. An expiration time specified in the definition of a transaction becomes the default expiration period for that transaction type and applies to all instances of the transaction. An expiration time specified in an OTMA message header applies only to the transaction instance submitted with the OTMA header and overrides any expiration time specified in the transaction definition.

An expiration time specified in an OTMA message header can be specified as a point in time or as a length of time. If specified as a point in time, the expiration time is specified in store clock (STCK) format and represents the time of day at which a transaction expires. If specified as a length of time, the expiration time is specified in seconds and OTMA calculates the expiration time from the time at which OTMA receives the transaction from the z/OS cross-system coupling facility (XCF).

OTMA checks if a transaction is expired at three points:

1. When OTMA first receives a transaction from XCF. If the expiration time has already passed, OTMA discards the transaction and returns a NAK message to the client.
2. When OTMA enqueues a transaction in IMS. If a transaction expires before OTMA enqueues the transaction, OTMA discards the transaction and returns a NAK message to the client.
3. For transactions that are not MSC, Fast Path, or conversational, when an IMS application program issues a GU call to retrieve a transaction from the input queue.

   If a transaction expires on the input queue before an IMS application program can retrieve it with a GU call, OTMA discards the transaction and, by default, issues message DFS3688I to the OTMA client. However, if the TMAMINPT flag is set in the OTMA input message that expires, instead of the returning a DFS3688I message, OTMA sends back the original input transaction data to the OTMA client.

   For each OTMA client, you can optionally configure OTMA to create a symptom dump and issue a DFS554A message for transactions that expire on the input queue by specifying TODUMP=YES in the OTMA client descriptor in the DFSYDTx member of the IMS.PROCLIB data set.

   You can also request a symptom dump and DFS554A message for individual messages by setting the TMAMDUMP flag in the OTMA state data prefix of input messages. Setting the TMAMDUMP flag on an individual message overrides a specification of TODUMP=NO for that message only.

For the following transaction types, OTMA monitors the expiration time only until the transaction is enqueued in IMS:

- MSC remote transactions
- Fast Path transactions

OTMA does not monitor the expiration time for the following transaction types:

- IMS conversational transactions
- Transactions queued as a result of a program-to-program switch

When you define an expiration time period in the definition of a transaction, you use the EXPRTIME parameter of either the TRANSACT stage-1 system definition macro or the type-2 dynamic resource definition commands **CREATE TRAN** or **UPDATE TRAN**. You can also specify an expiration time period for transactions created by the Destination Creation exit routine (DFSINSX0). Transaction expiration times enabled by any of these methods are not specific to OTMA.

Specifying OTMA transaction expiration time in STCK format

You can specify a transaction expiration time in store clock (STCK) format in the OTMA message header.

When specified in STCK format, the expiration time is the time of day that the transaction expires. If the transaction is either not retrieved by an IMS application program or enqueued for MSC or conversational processing before the expiration time of day, the transaction expires and OTMA discards the transaction.

To specify an expiration time period in STCK format for an OTMA transaction:

- In the state data section of the OTMA header, specify X'01' in the TMAMHIST field.
- In the state data section of the OTMA header, specify the byte offset of the time specification in the TMAMOSXP field. This is the offset of the time specification in the user data section of the OTMA header.
- In the user data section of the OTMA header, code the OTMA client to specify in STCK format the time of day at which the transaction expires.

Related reference

“Transaction and callout messages” on page 874
The state data for the transaction-related and synchronous callout request information in the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

“User data section” on page 886
The user-data section of the OTMA message prefix is variable length and follows the security-data section. It can contain any data.

Specifying OTMA transaction expiration time in seconds

You can specify a transaction expiration time as a length of time in seconds in the OTMA message header.

When specified in seconds, OTMA calculates the expiration time by adding the length of time specified to the time at which OTMA receives the transaction from the z/OS cross-system coupling facility (XCF).

For example, if the length of time specified is 10 seconds, the transaction expires 10 seconds after OTMA receives it from XCF. If the transaction is neither retrieved by an IMS application program nor enqueued for MSC or conversational processing before 10 seconds pass, the transaction expires and OTMA discards the transaction.

To specify a length of time for transaction expiration, code the following fields in the state data section of the OTMA header:

- In the TMAMHIST field, specify X'02'.
- In the TMAMOSXP field, specify the length of time in seconds that OTMA uses to calculate the expiration time. OTMA calculates the expiration time by adding this value to the time of day that OTMA receives the transaction from XCF.

Related reference

“Transaction and callout messages” on page 874

The state data for the transaction-related and synchronous callout request information in the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

Specifying the number of SAPs IMS allocates for OTMA input messages

To maintain performance and limit the usage of ECSA storage, you can adjust the number save area prefixes (SAPs) that IMS pre-allocates for each OTMA client and the maximum number of SAPs that IMS can allocate dynamically during run time by specifying the DSAP and DSAPMAX parameters in the OTMA DFSYDT*x* member of the IMS PROCLIB data set.

IMS uses the SAPs with the OTMA input message (OIM) task control block (TCBs) of an OTMA client to process OTMA input messages. IMS requires one SAP for each input message. After IMS finishes processing a message, IMS reuses the SAP for another message.

By default, IMS pre-allocates 18 SAPs for an OTMA client when the client connects. The pre-allocated SAPs remain allocated to the OTMA client until IMS is restarted.

If all SAPs are in use when new input messages are received from an OTMA client, IMS dynamically allocates more SAPs as needed until the maximum number of SAPs is reached. By default, the maximum number of SAPs that can be dynamically allocated to an OTMA client is 500. After dynamically allocated SAPs are no longer needed, IMS gradually cleans them up, until only the pre-allocated SAPs remain.

If IMS cannot dynamically allocate more SAPs quickly enough or the maximum number of SAPs are already allocated, IMS uses selective dispatching to prioritize processing so that higher priority work is processed before lower priority work.

During IMS system checkpoints, if IMS used selective dispatching for OTMA input messages for any length of time since the last system checkpoint, IMS issues message DFS0769I, OIM SELECTIVE DISPATCHING - SAPS.

In most cases, selective dispatching is active only momentarily until IMS allocates more SAPs and does not indicate a problem nor significantly impact performance. However, if IMS uses selective dispatching frequently or you expect regular high volumes of critical work, you might consider adjusting the number of SAPs that IMS pre-allocates to ensure that there are enough to process all incoming OTMA messages. You might also consider increasing the maximum number of SAPs that IMS can allocate to OTMA clients.

Statistics for the usage of SAPs by OIM TCBs are written at an IMS level in the x'450F' dispatcher statistics log record. They are written at an individual OTMA member level in the X'4518' individual TCB dispatcher statistics log record. Examining the statistics in these log records can provide useful information about the occurrences of selective dispatching.

IMS stores SAPs in the extended common storage area (ECSA). Allocating more SAPs uses more ECSA storage.

- To specify the number of SAPs that IMS pre-allocates for an OTMA client, specify the DSAP parameter in the OTMA client descriptor in the DFSYDT*x* member of the IMS PROCLIB data set.
- To specify the maximum number of SAPs that IMS can allocate dynamically for an OTMA client, specify the DSAPMAX parameter in the OTMA client descriptor in the DFSYDT*x* member of the IMS PROCLIB data set.
- To specify the OTMA system default value for the number of pre-allocated SAPs, specify the DSAP parameter in the DFSOTMA descriptor in the DFSYDT*x* member of the IMS PROCLIB data set.
- To specify the OTMA system default value for the maximum number of SAPs that IMS can allocate dynamically, specify the DSAPMAX parameter in the DFSOTMA descriptor in the DFSYDT*x* member of the IMS PROCLIB data set.

Related reference

OTMA client descriptor syntax and parameters (System Definition)

DFSOTMA descriptor syntax and parameters (System Definition)

Related information

DFS0769I (Messages and Codes)

Terminating conversational transactions in OTMA

Occasionally, you might need to terminate a conversational transaction in OTMA, either to cleanup a message on a tpipe queue after a conversation hangs or for some other reason.

A conversational transaction might hang, for example, if the OTMA client that initiated the conversation terminates without IMS detecting the termination. In such a case, the storage associated with the hung transaction, such as the storage required for a transaction instance block (TIB) and a scratchpad area, is not freed. Over time, if hung transactional conversations are not terminated, the amount of storage that is wasted on them can become significant.

You can use one of the following options to terminate hung conversational transactions.

- Specify the **ENDCONV=** parameter of the DFSOTMA descriptor in the DFSYDT*x* member of the IMS PROCLIB data set.

  If the conversational transaction remains idle for the specified period of time after the prior iteration of the conversational transaction completes, OTMA ends the transaction and releases IMS resources that were allocated for the transaction.
- Use the **QUERY OTMATI** and **/EXIT CONV** commands:
  a) Issue the IMS type-2 command **QUERY OTMATI** specifying the SHOW(ALL) keyword and the name of one or more tmembers, tpipes, or both.
  b) In the output of the **QUERY OTMATI** command in the CONVID column, note the conversational ID of the transaction instances that have a message age value in the MsgAge column that is large enough to indicate a hung conversational transaction.
  c) After identifying the transaction instance and noting its conversational ID, issue the **/EXIT CONV** *conv_id* TMEMBER *tmembername* TPIPE *tpipename* command.

Related reference

/EXIT command (Commands)

QUERY OTMATI command (Commands)

DFSOTMA descriptor syntax and parameters (System Definition)

OTMA security

Security for OTMA is enforced by RACF or a similar security product. The following topics describe implementing security for OTMA and OTMA clients by using RACF.

RACF security levels for OTMA

OTMA uses four levels of RACF security: CHECK, FULL, JOIN, NONE, PROFILE.

Only one RACF security level can be in effect at one time for OTMA globally; however, using the /SECURE OTMA TMEMBER command, you can specify different security levels for individual OTMA clients.

NONE

A system-wide security level. RACF is not called for messages received through OTMA. Specifically:

RACF is not called when IMS receives the connection request (client-bid) from IBM MQ or IMS Connect.

RACF is not called to verify that the user ID in the incoming message is a valid user ID (one that has been defined to RACF).

RACF is not called to verify that the user ID in the incoming message is authorized to the IMS command or IMS transaction requested in the message.

The user ID caching scheme is not used.

PROFILE
A message-by-message security level. In other words, each incoming message entered through OTMA is checked to determine whether or not RACF will be called. IMS checks each incoming message independently to see if the security value is set to NONE, CHECK, or FULL. Specifically:

Messages entered from IMS Connect will contain a 1-byte security flag field. The value in this field determines whether or not RACF is called.

Messages entered from the IBM MQ-IMS Bridge application will contain a SecurityScope field in the MQIIH structure. The value in this field will determine whether or not RACF is called.

Tip: Consider using the PROFILE security level for situations when application developers set the RACF security level as N (NONE), C (CHECK), or F (FULL) in each incoming message. In this case, the security level set in each message determines whether IMS calls RACF for security checking related to that message. You might not want application programmers deciding on the security for IMS commands and IMS transactions. RACF is called when IMS receives the connection request (client-bid) from IBM MQ or IMS Connect.

JOIN
A system-wide security level, which means that RACF is called only to authorize the user ID on the initial client bid request from an OTMA client, such as IMS Connect or MQ series.

After the connection is authorized, no additional transaction or command security checking is performed on messages that are received on the connection.

CHECK
A system-wide security level, which means that RACF is called for messages received through OTMA. Specifically, RACF is called:

- For client-bid connection requests. A cache, or hash table, is built for each OTMA client if the client-bid is successful.

  A user ID caching scheme is used in IMS/OTMA environments. The caching scheme also improves authorization checking performance.

  A cache, or hash table, is used to store previously verified user IDs. Each OTMA client (IMS Connect, IBM MQ for z/OS, or others) has a hash table created in the IMS control region after a successful client bid. Use of the hash table minimizes the number of calls to RACF to VERIFY user IDs. This way, if the same user ID enters multiple messages destined for IMS/OTMA, IMS can check the hash table for a valid entry for the user ID and might be able to avoid the VERIFY call to RACF. The entry for the user ID in the hash table contains a pointer to the accessor environment element (ACEE) for the user ID. The ACEE that is pointed to can be used for resource (command and transaction) FASTAUTH calls to RACF.
- To VERIFY that the user ID in the incoming message is a valid user ID (one that has been defined to RACF).

  If the OTMA client (IMS Connect or IBM MQ for z/OS) supplied a UTOKEN in the incoming message, IMS supplies the address of the UTOKEN on the VERIFY call to RACF. Use of the UTOKEN in VERIFY processing improves performance. RACF returns an ACEE security control block to IMS for verified user IDs.
- To verify that the user ID in the incoming message is authorized to the IMS command or IMS transaction requested in the message. The address of the ACEE, previously built by RACF during verify authorization processing, is supplied by IMS on the FASTAUTH call to RACF.
- To verify that the user ID in the incoming message is authorized to the IMS transaction code set as the destination on a DL/I CHNG or AUTH call. A cached ACEE is used for these calls, which eliminates performance concerns for application programs that issue many CHNG or AUTH calls.

FULL
A system-wide security level, which means that RACF will be called for messages received through OTMA.

FULL has the same characteristics as CHECK, with the following exceptions:

- During the verify processing, RACF is called a second time to build an additional ACEE security control block in the dependent region.
- The dependent region runs under the requestor's user ID. Any resources, such as files, that are accessed from the dependent region must be authorized to that user ID.

Specifying OTMA security

You can make most security specifications for OTMA and OTMA clients by using the transaction message prefixes, the **/SECURE OTMA** command, and during system definition.

Not every method of specifying OTMA security has the same scope of affect. Some methods allow you to specify security for all members of the OTMA z/OS cross-system coupling facility (XCF) group, and other methods allow you to specify security for only an OTMA client or an OTMA transaction or command message.

Specifying OTMA security during system definition

You can use the OTMASE= execution parameter in the IMS and DCC execution procedures to set the RACF security level globally for all the OTMA clients in an OTMA z/OS cross-system coupling facility group. If you use IMS Connect with OTMA, you can also enable RACF statistics to be recorded when OTMA client connections to IMS TM are authenticated.

- The levels of RACF security that you can specify by using the OTMASE= parameter include:
  - CHECK
  - FULL
  - JOIN
  - NONE
  - PROFILE

  Specifications made using the OTMASE= execution parameter can be overridden by issuing the **/SECURE OTMA** command.
- If you use IMS Connect with OTMA and you want to enable, during system definition, RACF statistics to be recorded when OTMA client connections to IMS TM are authenticated, specify TMRACFST=Y in the HWS statement of the HWSCFGxx member of the IMS PROCLIB data set.

Related tasks

“Defining the level of OTMA security checking” on page 769
If you use a security product such as RACF, you can specify different levels of security checking by using the OTMASE parameter.

“Enabling RACF security statistics for IMS Connect” on page 177
If IMS Connect is configured to call RACF, you can enable RACF security statistics to be recorded and updated when IMS Connect issues the RACF call **RACROUTE REQUEST=VERIFY**. You can enable RACF statistics to be recorded and updated for ODBM client connections to IMS DB and for OTMA client connections to IMS TM. After you enable RACF statistics, the statistics are updated no more than once per day.

Related reference

“RACF security levels for OTMA” on page 806
OTMA uses four levels of RACF security: CHECK, FULL, JOIN, NONE, PROFILE.

Procedures used in IMS environments (System Definition)

ODACCESS statement (System Definition)

QUERY IMSCON TYPE(CONFIG) command (Commands)

Modifying OTMA security online

You can modify security for all OTMA clients globally and for OTMA clients individually by using online commands. You can also specify online whether RACF statistics are recorded when IMS Connect issues the RACF call **RACROUTE REQUEST=VERIFY** to authenticate OTMA client connections to IMS TM.

Modifying security for all OTMA clients globally and for OTMA clients individually

You can modify security for all OTMA clients globally and for OTMA clients individually by using the online type-1 command **/SECURE OTMA**. You can use the **/SECURE OTMA** command to override security specifications made by using the OTMASE= parameter in the IMS or DCC execution procedures.

The **/SECURE OTMA** command can specify the RACF security level globally for the OTMA z/OS cross-system coupling facility (XCF) group or disable RACF security for the OTMA XCF group by using the following parameters:

- **/SECURE OTMA CHECK**
- **/SECURE OTMA FULL**
- **/SECURE OTMA JOIN**
- **/SECURE OTMA PROFILE**
- **/SECURE OTMA NONE**

You can specify a RACF security level for individual OTMA clients by issuing the following commands:

- **/SECURE OTMA CHECK TMEMBER** *tmember_name*
- **/SECURE OTMA FULL TMEMBER** *tmember_name*
- **/SECURE OTMA JOIN TMEMBER** *tmember_name*
- **/SECURE OTMA PROFILE TMEMBER** *tmember_name*
- **/SECURE OTMA NONE TMEMBER** *tmember_name*

Security specifications made for individual OTMA clients override the global security settings made for the rest of the OTMA XCF group.

If you specify **/SECURE OTMA NONE**, IMS does not use RACF for security verification, regardless of what security is specified by the class for a client-bid request or for transactions.

When RACF security checking is disabled for OTMA, you can issue only the following default IMS commands through OTMA:

- **/BROADCAST**
- **/LOCK**
- **/LOG**
- **/RDISPLAY**
- **/UNLOCK**

Complete information for how to use these command is provided in *IMS Version 15 Commands, Volume 1: IMS Commands A-M* and *IMS Version 15 Commands, Volume 2: IMS Commands N-V*.

Enabling RACF statistics for OTMA client connections to IMS TM

To enable, online, RACF statistics to be recorded when IMS Connect issues the **RACROUTE REQUEST=VERIFY** call to authenticate OTMA client connections to IMS TM, use the TMRACFST(ON) keyword on the **UPDATE IMSCON TYPE(CONFIG)** command.

After you enable RACF statistics, IMS Connect uses the STAT=ASIS parameter on the **RACROUTE REQUEST=VERIFY** call. With STAT=ASIS, the RACF messages and statistics are controlled by the installation's current options on the RACF **SETROPTS** command.

After you enable RACF statistics, the statistics are recorded by RACF no more than once per day to a system management facility (SMF) data set or log stream. The SMF data set or log stream that is used to record the RACF statistics is specified in the RACF configuration.

Related tasks

"Enabling RACF security statistics for IMS Connect" on page 177
If IMS Connect is configured to call RACF, you can enable RACF security statistics to be recorded and updated when IMS Connect issues the RACF call **RACROUTE REQUEST=VERIFY**. You can enable RACF statistics to be recorded and updated for ODBM client connections to IMS DB and for OTMA client connections to IMS TM. After you enable RACF statistics, the statistics are updated no more than once per day.

Related reference

ODACCESS statement (System Definition)
QUERY IMSCON TYPE(CONFIG) command (Commands)
UPDATE IMSCON TYPE(CONFIG) command (Commands)

Refreshing RACF ACEEs for OTMA

When OTMA security is enabled, for performance purposes OTMA caches in memory RACF user profiles as a RACF accessor environment element (ACEE).

The ACEE is then used in subsequent calls to RACF to determine the user ID's authorization to the IMS command or IMS transaction requested in the input message. Using the ACEE reduces I/O to RACF.

When changes are made to a user profile in the RACF database on DASD, the changes are not reflected in the cached ACEE if IMS fails to register with RACF event notification facility (ENF), and the user's old access privileges might remain unchanged in online memory until the ACEE is refreshed.

RACF ACEEs that are cached by OTMA can be refreshed with one of the following methods:

- Automatically by OTMA when the z/OS ENF notifies IMS of the changes.

  When changes are made to a user profile in the RACF database on DASD, ENF notifies IMS of the changes with an ENF event code 71. Upon receiving the notification, OTMA refreshes the ACEEs for the changed user ID.

  IMS automatically registers with ENF during startup to receive the event code 71 notifications. If ENF registration fails for any reason, IMS issues message DFS3525E and OTMA cannot refresh ACEEs automatically when changes are made in RACF.

  To ensure that cached ACEEs reflect the latest RACF security definitions even when IMS cannot register with ENF, OTMA clients can specify an aging value that defines the length of time between automatic refreshes of cached ACEEs.

  If ENF registration failed during IMS startup and your RACF security administrator modifies a user's access privileges for OTMA in RACF, you might need to refresh the ACEE for the user ID by issuing the **/SECURE OTMA REFRESH** command. Otherwise, the user's old access privileges might remain unchanged in online memory until the ACEE aging limit is reached.
- You can specify an aging value for OTMA ACEEs. When the OTMA ACEE aging value is reached, OTMA refreshes the ACEE before it processes the next input message that is received from an OTMA client.

  To define an ACEE aging value for IMS Connect to pass to OTMA, use the OAAV= keyword in the DATASTORE configuration statement of the IMS Connect PROCLIB member, HWSCFGxx.

  To define an ACEE aging value for IBM MQ, use the OTMACON= parameter in the CSQ6SYSP macro.

  To define a global ACEE aging value for all OTMA clients, issue the **/SECURE OTMA ACEEAGE** command. If you define a global ACEE aging value for OTMA clients by issuing the **/SECURE OTMA ACEEAGE** command, the aging value that you specify overrides the existing ACEE aging values for all OTMA clients. If you specify the **TMEMBER** parameter with the **/SECURE OTMA ACEEAGE** command, the ACEE aging value that you define overrides the existing value, if any, for the OTMA client that you specify.

Each cached OTMA ACEE can have its own aging value that is based on the OTMA client (TMEMBER) with the lowest aging value that accesses it. For example, IBM MQ sets its ACEE aging value to five days and IMS Connect sets its ACEE aging value to one day. Any ACEEs that only IBM MQ uses have an aging value of five days. Any ACEEs that only IMS Connect uses have an aging value of one day. If both IBM MQ and IMS Connect use an ACEE, the aging value is one day, which is the lowest value between five days and one day.

The ACEE expiration value is specified during the client-bid time.

- You can issue the **/SECURE OTMA REFRESH** command to manually refresh the OTMA ACEEs.

  To refresh all OTMA ACEEs globally, issue the command **/SECURE OTMA REFRESH**. To refresh the ACEE for an individual OTMA client, issue the command **/SECURE OTMA REFRESH TMEMBER** *tmember_name*. To refresh only those ACEEs for a specific user ID, issue the command **/SECURE OTMA REFRESH USER** *user_id*.
- You can also rebuild the ACEE table by issuing the **/STOP** and **/START OTMA** commands.

Related tasks

“OTMA ACEE flood control” on page 800
The OTMA accessor environment element (ACEE) flood control function prevents virtual storage in the IMS control region from running out.

Related reference

/SECURE command (Commands)

DFSOTMA descriptor syntax and parameters (System Definition)

HWSCFGxx member of the IMS PROCLIB data set (System Definition)

Security specifications in OTMA message prefixes

Security specifications in the OTMA message prefix are in the security-data section. The security-data section is mandatory for every transaction or command and is optional for OTMA protocol commands.

The specifications you can make for security in an OTMA message prefix include:

- The RACF security levels to be used with this message, including:
  - FULL
  - CHECK
  - NONE
- The user token
- The user ID
- The SAF profile
- The network user ID
- The network session ID

In the state data of a client-bid request, you can specify an accessor environment element (ACEE) aging interval. An ACEE aging interval represents the amount of time an ACEE can be used before being refreshed. If the aging interval has expired when an input message is received by OTMA, OTMA refreshes the ACEE before validating the message.

Related reference

“OTMA message prefix” on page 859

OTMA messages have a prefix that conforms to a format that is mapped by the DFSYMSG macro in the IMS.ADFSMAC data set.

RACF security classes used by OTMA

To have RACF secure transactions or commands submitted to IMS through OTMA, you must define the security to be enforced for them to RACF.

Transaction security definitions are stored in the RACF TIMS class. Command security definitions are stored in the CIMS class. If a transaction is not in the TIMS class or a command is not in the CIMS class, the transaction or command is allowed regardless of any options you might specify by using the **/SECURE OTMA** command.

When you enter an OTMA command, OTMA issues a RACHECK to validate the command. OTMA passes only the command verb to DFSCCMD0 for verification, not the entire CVB control block.

If you are using RACF to secure asynchronous hold queues from unauthorized users of the RESUME TPIPE call, you must define either the default RIMS resource or a R*xxxxxxx* resource class, where *xxxxxxx* is the value of the RCLASS parameter in the DFSDCxxx PROCLIB member. The RACF resource class that you define must include the names of the protected asynchronous hold queues and the user IDs that are authorized to access each queue.

Distributed network security credential support and OTMA

OTMA supports security credentials that are entered by a user in a distributed environment. After distributed network security credentials are passed to OTMA, the credentials are included in IMS log records that contain information about the message prefix.

The distributed network security credentials can include a network user ID and a network session ID.

Network user ID
: The distributed identity of the user. The maximum length of a network user ID is 246 bytes. For users of the IMS TM Resource Adapter, the network user ID is a Distinguish Name (DN) in the X.500 series of standards.

Network session ID
: The session identity of the distributed user. The maximum length of a network session ID is 254 bytes. For users of the IMS TM Resource Adapter, the network session ID is a domain name, realm, or registry name.

Network security credentials can be passed to IMS from applications in a distributed environment that use one of the following user message exits:

- HWSSMPL0
- HWSSMPL1
- HWSJAVA0

Restriction: Distributed network security credentials from DataPower, IMS Connect API, and SOAP Gateway clients are not supported by IMS Connect.

To enable network security credentials to be passed from user-written applications that use the HWSSMPL0 or the HWSSMPL1 user message exit, you must define IRM extensions, which are used to pass the network security credentials that are entered by a user to IMS Connect. An IRM extension with an ID of *NETUID* is used to pass a network user ID and an IRM extension with an ID of *NETSID* is used to pass a network session ID. After the network security credentials are passed to IMS Connect, the HWSSMPL0 and HWSSMPL1 user message exit routines use the network security credentials that are in the IRM extensions to build the OTMA message prefix. The security credentials are included in the security-data section of the OTMA message prefix.

When a user enters network security credentials to an application that uses the HWSJAVA0 user message exit, the credentials are passed from the application to IMS TM resource adapter, which then includes the credentials in the security-data section of the OTMA message prefix.

Because distributed network security credentials are passed to IMS in the security-data section of the OTMA message prefix, all IMS log records that contain information about the message prefix, such as log records X'01' and X'03', include the distributed security credentials.

If a Fast Path message contains network security credentials and is processed by the Fast Path expedited message handler (EMH) on the local IMS system, the credentials are logged in the X'5901' log record.

If a Fast Path message that contains network security credentials is processed by using the EMH queue (EMHQ) in a shared-queues environment, in the front-end IMS system, the credentials are included in the X'5911' log record. In the back-end IMS system, which is the processing IMS system, the credentials are included in the X'5901' log record.

You can enable IMS to log distributed network security credentials in RACF SMF records after the credentials are passed to OTMA. To enable distributed network security credentials to be logged in RACF SMF records, specify LOGSTR=YES in the OTMA client descriptor of the DFSYDTx member of the IMS PROCLIB data set. After LOGSTR=YES is specified, the first 255 bytes of the distributed network security credentials that are sent to OTMA are logged in RACF SMF records.

You can use the `otma_send_receivey` and `otma_send_asyncx` APIs of the IMS OTMA Callable Interface (OTMA C/I) to pass the network user ID and the network session ID to IMS. For each API, up to 100 bytes for the network user ID and up to 100 bytes for the network session ID can be passed to IMS.

You can also use the Transaction Authorization exit routine (DFSCTRN0) to pass the addresses of the network security credentials in the OTMA message prefix.

You can use the following OTMA user exit routines, which include the address of the security-data section of the OTMA message prefix, to access the network security credentials that are in the OTMA input message if the credentials are passed to IMS:

- DFSYIOE0
- DFSYPRX0
- DFSYDRU0

Related reference

otma_send_asyncx API (Application Programming APIs)

otma_send_receivey API (Application Programming APIs)

“Security data section” on page 883
The security-data section is mandatory for every transaction or command, and is optional for OTMA protocol commands.

“Explanation of OTMA security data fields” on page 884
The following information provides additional detail on the content of the security-data section of the message prefix.

OTMA Input/Output Edit user exit (DFSYIOE0 and other OTMAIOED type exits) (Exit Routines)

OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits) (Exit Routines)

OTMA User Data Formatting exit routine (DFSYDRU0) (Exit Routines)

Transaction Authorization exit routine (DFSCTRN0) (Exit Routines)

OTMA client descriptor syntax and parameters (System Definition)

Securing messages on the asynchronous hold queue

You can protect messages on asynchronous hold queues from unauthorized use of the RESUME TPIPE call by using either RACF, the OTMA Resume TPIPE Security user exit (OTMARTUX), or both.

When security is enabled for tpipe hold queues, the user ID issuing the RESUME TPIPE call must be authorized to access the TPIPE name that is contained in the RESUME TPIPE call message before any of messages are sent to an OTMA client.

The security checking performed by RACF and the security checking performed by the OTMARTUX user exit are optional. They can be used in combination or either one by itself. If both RACF and the OTMARTUX

are used, RACF is called first before giving control to the OTMARTUX user exit, in which case, the OTMARTUX user exit can override RACF, depending on your needs.

Securing asynchronous hold queues by using RACF

When RACF security checking is enabled for an asynchronous hold queue, the authorization logic verifies and validates the security header and authorizes the user ID under the TPIPE name.

When a Resume TPIPE call is received, RACF security checking is performed only if a Resume TPIPE resource class (RIMS or Rxxxxxxx) exists and the tpipe name specified on the call is defined in the resource class.

Regardless of whether RACF security is enabled, you can use the OTMA Resume TPIPE Security user exit (OTMARTUX). If both the OTMARTUX user exit and RACF are used, the RACF security is always called first. In such a case, the OTMARTUX user exit can override the results of the RACF procedure.

To enable Resume TPIPE security:

1. If one does not already exist, define a Resume TPIPE resource class (RIMS or Rxxxxxxx) by using the RCLASS keyword on the SECURITY system definition macro.

   During IMS startup, if no Resume TPIPE resource class is defined to IMS, IMS issues message DFS3187I. After IMS is running, no further warnings are issued to alert you to the absence of a Resume TPIPE resource class.
2. In the Resume TPIPE resource class, specify the tpipe names of the asynchronous hold queues to be protected and the user IDs that are authorized to access the queues.
3. Specify an appropriate level of RACF security for OTMA by using either the OTMASE startup parameter or the **/SECURE OTMA** command.

   The appropriate levels of OTMA security that you can specify are FULL, CHECK, or PROFILE. If a security level of PROFILE is specified, the resume tpipe request message must specify either FULL or CHECK.
4. Code the resume tpipe request messages to access the RACF-protected asynchronous hold queues. The resume tpipe request messages must include:
   - The tpipe name in the control data section of the OTMA prefix
   - A user ID in the security section of the OTMA prefix
   - If the OTMA security level is PROFILE, a security flag setting of either FULL or CHECK

Related information

DFS3187I (Messages and Codes)

Securing asynchronous hold queues by using the OTMA Resume TPIPE Security user exit (OTMARTUX)

The OTMA Resume TPIPE Security user exit (OTMARTUX) provides one of two possible layers of security for RESUME TPIPE calls issued to retrieve message queued to the OTMA asynchronous hold queue.

When security for RESUME TPIPE calls is enabled, the OTMARTUX user exit checks the caller's authority when the RESUME TPIPE call is initiated, but before retrieving the messages from the hold queue.

You can use OTMARTUX user exit either with or without RACF security checking for the RESUME TPIPE call. If both the OTMARTUX user exit and RACF are used, the RACF security is always called first. In such a case, the OTMARTUX user exit can override the results of the RACF procedure.

When security for the RESUME TPIPE call is enforced by both RACF and the OTMARTUX user exit, the OTMARTUX user exit is invoked regardless of the success or failure of the RACF security procedure. The OTMARTUX user exit can accept the results of the RACF security check, override the results, or enforce more restrictive security rules. An example of a more restrictive rule might be to authorize a user to access the output messages only within a specific period of time during the day.

When authorization is successful, output messages in the hold queue are returned to IMS Connect.

When authorization fails, a rejection message (NAK) of the RESUME TPIPE call is sent to the client.

To bypass the OTMARTUX user exit, ensure that your RESLIB does not contain DFSYRTUX and do not define an EXITDEF statement for the OTMARTUX user exit type in the USER_EXITS section of your DFSDF*xxx* member.

Related reference

OTMARTUX: OTMA Resume TPIPE Security user exit (DFSYRTUX and other OTMARTUX type exits) (Exit Routines)

Security for OTMA IMS-to-IMS TCP/IP connections

For OTMA ALTPCB messages sent to a remote IMS system on an IMS-to-IMS TCP/IP connection, transaction authorization is performed by the remote IMS system.

When an application program running in the local IMS system sends a message to a remote system by issuing an ISRT ALTPCB call, the user ID of the application program is included in the prefix of the message.

You can also specify a user ID in the OTMA destination descriptor in the OTMA DFSYDTx member of the IMS.PROCLIB data set. If a user ID is specified in an OTMA destination descriptor, the remote IMS system uses the user ID in the OTMA destination descriptor instead of the user ID of the application program that issued the ISRT call.

You can secure the TCP/IP connection by implementing RACF PassTicket user authentication in IMS Connect. IMS Connect authenticates a RACF PassTicket when a connection is first established. When persistent sockets are used, after the initial authentication is performed, all messages received on the connection are treated as coming from a trusted user and no further authentications is performed as long as the connection persists.

Related tasks

“Securing IMS-to-IMS TCP/IP connections” on page 181
To secure IMS-to-IMS TCP/IP connections, IMS Connect uses RACF PassTickets to establish one instance of IMS Connect as a trusted user of another instance of IMS Connect.

IMS-to-IMS TCP/IP connections (System Definition)

Related reference

DFSYDTx member of the IMS PROCLIB data set (System Definition)

General OTMA security considerations

A number of general security considerations exist for OTMA that you should be aware of.

- If you use RACF (or an equivalent product) for security, define the `IMSXCF.group.client_member_name` in the FACILITY class.

  If you define the `IMSXCF.group.client_member_name` in the FACILITY class, and if IMS security is not set to `NONE`, the user token for the client-bid request must be valid and the user must have READ access to the FACILITY class.

  If the user token for a client-bid request fails RACF verification, the client receives a NAK message from the server.
- Authorize the z/OS cross-system coupling facility client for z/OS.
- If you define your OTMA applications with full security, the security environment is kept until the application ends.
- After IMS receives messages from OTMA, when OTMA security is activated, IMS calls RACF to verify that the user ID in the incoming message is a valid RACF user ID. IMS is not passed a password for the user ID, so the call to RACF is to verify the user ID only. If a password must be validated, it must be validated before sending the message to IMS.
- IMS uses the UTOKEN in the input message in the call to RACF not only to verify the user ID, but also to create a security control block in the IMS control region to represent each verified user ID. The security

control blocks built in the IMS control region, representing verified RACF user IDs, are called accessor environment elements or ACEEs.

- The DL/I ICAL call has special security requirements related to the OTMA security configuration. The DFSYICAL tmember that is used for synchronous program switch request processing uses OTMA security configuration settings even if OTMA is not enabled.
- If distributed network security credentials, including the network user ID and the network session ID, are passed to IMS in the security-data section of the OTMA message prefix, the credentials are included in IMS log records, such as X'01' and X'03'. To enable distributed network security credentials to be logged in RACF SMF records, specify LOGSTR=YES in the OTMA client descriptor of the DFSYDTx member of the IMS PROCLIB data set. After LOGSTR=YES is specified, the first 255 bytes of the distributed network security credentials that are sent to OTMA are logged in RACF SMF records.

Related concepts

"Distributed network security credential support and OTMA" on page 812
OTMA supports security credentials that are entered by a user in a distributed environment. After distributed network security credentials are passed to OTMA, the credentials are included in IMS log records that contain information about the message prefix.

IMS security (System Administration)

Related reference

ICAL call (Application Programming APIs)

Using DL/I calls in an OTMA environment

Certain DL/I calls have special considerations when used with OTMA.

CHNG

If a CHNG call is issued from an OTMA submitted transaction, the destination is assumed to be the same OTMA client (the tpipe name is set by the CHNG call). This behavior can be altered by the OTMA Prerouting and Destination Resolution exit routines.

An IMS application program that issues a CHNG call to an alternate PCB (specifying an options list) does not cause IMS to call the OTMA Prerouting and Destination Resolution exit routines to determine the destination. However, an IMS application program that issues a CHNG call to an alternate PCB (specifying an APPC descriptor) does cause IMS to call the OTMA exit routines to determine the destination.

The application program can still issue ISRT calls to the I/O PCB to send data to an OTMA destination.

OTMA application programs can use CHNG and ISRT calls for APPC destinations.

INQY (null)

An INQY call issued for an OTMA destination returns the following information: the transaction pipe name, the client z/OS cross-system coupling facility member name, the user ID, the group name, and the synchronization levels.

ICAL

IMS application programs issue the ICAL call to send synchronous callout requests to a data or service provider that is external to the IMS installation. OTMA routes callout requests initiated by the ICAL call to a hold-queue capable OTMA client and routes the reply back to the waiting IMS application program. OTMA provides the ability to specify a timeout value in the OTMA destination descriptor for synchronous callout requests, which can be overridden by the a timeout value specified in the ICAL call.

The ICAL call can also be used to issue a synchronous program switch request. If the OTMA destination descriptor is configured with TYPE=IMSTRAN, OTMA switches control to another IMS application and waits for a response to send to the waiting IMS application program.

If the OTMA message prefix for a transaction from an OTMA client contains distributed network security credentials, the security credentials can be passed from IMS in synchronous callout requests that are initiated by the ICAL call. The distributed network security credentials are passed from IMS

via the ICAL call only if the RESUME TPIPE call is defined with the following field specifications in the IMS request message (IRM) prefix. If the following field specifications are not defined, IMS removes the distributed network security credentials from the security-data section of the OTMA message prefix in the synchronous callout request.

IRM_ARCH
X'05' (IRM_ARCH5)

IRM_F6
X'80' (IRM_F6_NWSE)

PURG
An IMS application program that issues a PURG call causes IMS to call the OTMA Prerouting and the Destination Resolution exit routines to determine the destination.

SETO
An IMS application program that issues a SET0 call does not cause IMS to call the OTMA Prerouting and the Destination Resolution exit routines to determine the destination.

Existing IMS application programs that issue SET0 calls might not run as expected because a return code is returned to the program if it is processing an OTMA-originated transaction. APPC/IMS application programs that issue SET0 calls might need modification if they require implicit OTMA support.

One way to make these application programs work is to use an INQY call before issuing the SET0 call. The application program can use the output from the INQY call to determine if a transaction originated from an OTMA client, and not issue the SET0 call.

For those DL/I calls that cause IMS to call one of the OTMA exit routines, IMS only calls the exit routines if the destination has not yet been set (for example, by another DL/I call).

To initiate protected conversations (such as accessing multiple resource managers' resources under one unit of recovery in an z/OS Resource Recovery Services environment), the client-adapter code (OTMA user) must acquire and own a private context and provide the context ID in the state-data section of the message prefix.

Definition: A *context* is a z/OS entity under which resource managers perform work; a private context is required in this environment.

During message traffic between IMS and the client, if the context-ID field in the message header is non-zero, protected conversation processing occurs.

Related tasks
Retrieving synchronous callout requests with RESUME TPIPE (Communications and Connections)
Related reference
Transaction management (Application Programming APIs)

OTMA program-to-program switch processing

Two types of message switch occur in OTMA: commit-then-send, and send-then-commit.

For OTMA commit-then-send input messages (also called asynchronous or commit mode 0 messages), the program switch always results in another commit mode 0 (CM0) message. For OTMA send-then-commit input messages (also called synchronous or commit mode 1 messages), the program switch results vary, depending on whether:

- there is an ISRT call to the I/O PCB
- an express PCB is used for the switch
- there is a switch to multiple programs
- the IMS start-up parameter OTMAASY=Y or OTMAASY=S is specified
- the transaction is protected

A P2P switch for a commit mode 1 (CM1) input message, therefore, could be another CM1 message, a DFS2082 message, or a CM0 message. In addition, some OTMA clients, for example IBM MQ, can accept a CM0 output message for a CM1 input message; others, however, may not.

The following topics provide usage scenarios for different send-then-commit message switches.

OTMA single-stream program switch

The single-stream program switch is shown in the following figure.

Program A switches to program B, and Program B switches to Program C, which then inserts back to the I/O PCB. This model of program flow delivers the send-then-commit output message successfully. Single-stream means that the program switches occur one after another. No express PCBs are used in the P2P message switches.

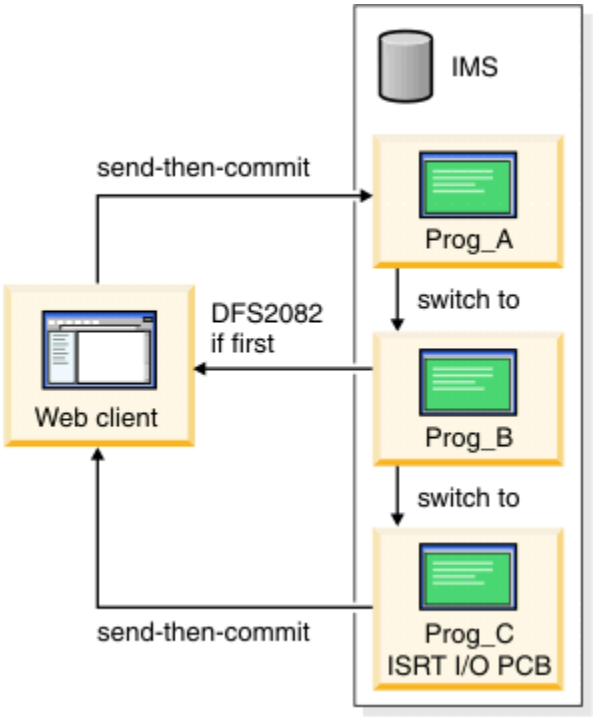


Figure 142. Single-stream program switch

OTMA program switch without ISRT to I/O PCB

When several switches occur in sequence and none inserts back to the I/O PCB, message DFS2082 is sent back to the OTMA client.

The last program switched to, Program C, does not insert back to the I/O PCB. IMS therefore generates message DFS2082 for the OTMA client.

Attention: If program C runs in a remote IMS through MSC and does not insert back to the I/O PCB, the remote IMS does not issue message DFS2082. However, in this case, the OTMA client program might hang and the front-end IMS control region will experience a build up of its control blocks. This kind of build up could result in a storage-related system outage. Restarting IMS releases the control blocks.

OTMA program switch with express PCB

A P2P message switch with an express PCB can lead to a commit-then-send output message.

Program A uses an express PCB rather than a non-express PCB to perform the P2P message switch. The output from Program B is commit-then-send because using the express PCB forces Program B to be processed asynchronously. When a program is processed asynchronously and inserts back to the I/O PCB, the output message is sent as a commit-then-send message. However, if Program A also switches to

Program C using a non-express PCB, Program C then inserts back to the I/O PCB. The output from C will be a send-then-commit message.

OTMA program switch to multiple programs

After a program inserts back to the I/O PCB, the rest of the program to program message switch, if any, is processed asynchronously.

For example, Program A switches to Program B, which inserts back to the I/O PCB. The output from Program B will be a send-then-commit message. Program B then switches to program C, which will be processed asynchronously.

A "race" condition can occur when a program switches to multiple programs. Program A switches to multiple programs using non-express PCBs. Only one switched-to program, the one scheduled first, is processed synchronously. The rest of the switched-to programs are processed asynchronously. If the program processed synchronously inserts back to the I/O PCB, the output message is a send-then-commit message.

In some cases, one of the multiple programs could be a remote program. This program flow is shown in the following figure. Program A switches to remote Program B through MSC. Program B first launches a new program, Program C, in the local IMS and then inserts a response to the OTMA client through the local IMS. Depending on what happens first (scheduling of Program C or the processing of the response for the OTMA client) in the local IMS, an unwanted DFS2082 message could be sent to the client. This is also a race condition. If Program C gets processed first in the local IMS, a DFS2082 message is sent. If the response is processed first, the expected output from Program B is delivered synchronously using send-then-commit.

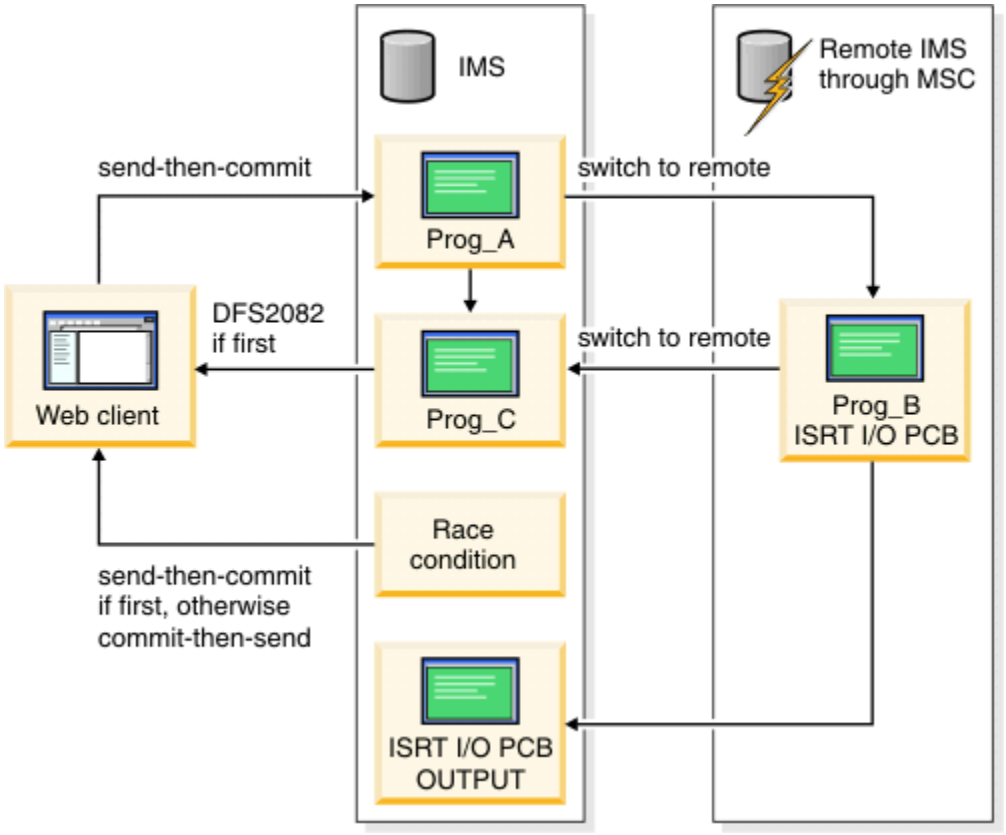


Figure 143. Race condition resulting from program switch to multiple programs

To avoid creating the race condition in these circumstances, you can do any one of the following:

- Modify your programs to avoid multiple program-to-program switches within the same transaction for send-then-commit (CM1).

- Use commit-then-send (CM0) input when performing multiple program-to-program switches within the same transaction.
- Use the IMS start-up parameter OTMAASY to serialize P2P message switch processing.

When you use the OTMAASY parameter to avoid a race condition, you can create a program switch model similar to the single-stream model. For example, in the preceding figure, Program B, for which the response mode transaction is processed synchronously, can deliver the send-then-commit (synchronous) output message. Program C, which is running with a non-response mode transaction, processes the message asynchronously.

Related tasks

"Specifying asynchronous delivery of program-to-program switch output messages" on page 772
If you are using send-then-commit (CM1) messages that initiate multiple program-to-program switches, to ensure that only the appropriate output is returned to the OTMA client in synchronous CM1 mode, specify OTMAASY=Y.

Related reference

Parameter descriptions for IMS procedures (System Definition)

OTMA program switch for protected transactions

For a non-conversational program that performs a P2P message switch for a protected transaction, ABENDU0711, with reason code 1D, will be returned. For a conversational program, the program receives an X6 status code.

Other OTMA program switch considerations

The following considerations also apply to program-to-program switching.

- The P2P message switch is not supported for OTMA protected messages (send-then-commit input with synclevel = SyncPt).
- If a non-conversational program performs a program-to-program message switch to a program in a Shared Queues environment, the program in the SQ environment must be running on the same IMS where the first program gets scheduled, unless the support for synchronous APPC/OTMA is active (AOS=Y) and the IMS start-up parameter OTMAASY=S is specified.
- If an input conversational transaction occurs, only the message-switched-to continuation of the conversation is scheduled synchronously. All other transactions are scheduled asynchronously.
- In a shared queues environment that has both synchronous APPC/OTMA support (AOS=Y on the DFSDCxxx PROCLIB member) and z/OS Resource Recovery Services (RRS) support (RRS=Y on the startup procedure) enabled, an application program running on a back-end IMS system that initiates an outbound APPC protected conversation with another IMS system is restricted to a single program-to-program switch to the same destination transaction.

  If an application program performs multiple program-to-program switches after allocating an APPC outbound protected conversation on another IMS system, the results are unpredictable and can include a WAIT-RRS/PC condition in the MPP dependent region.

Chapter 44. The OTMA client

The OTMA environment includes a server and one or more clients.

This chapter explains how a client interacts with the server to process IMS transactions.

What is an OTMA client?

An *OTMA client* is a z/OS application program that sends transactions to an IMS server and receives output. The application program must be a member of an z/OS cross-system coupling facility (XCF) group and use the OTMA protocol.

Heterogeneous (non-z/OS) networks can connect with z/OS in many ways. The following figure shows some of the possible applications that use XCF. These include:

- IBM MQ applications
- OEM applications
- IMS Connect applications
- DCE/RPC applications
- Other IBM applications

Any of these can connect to an OTMA client to communicate with IMS.

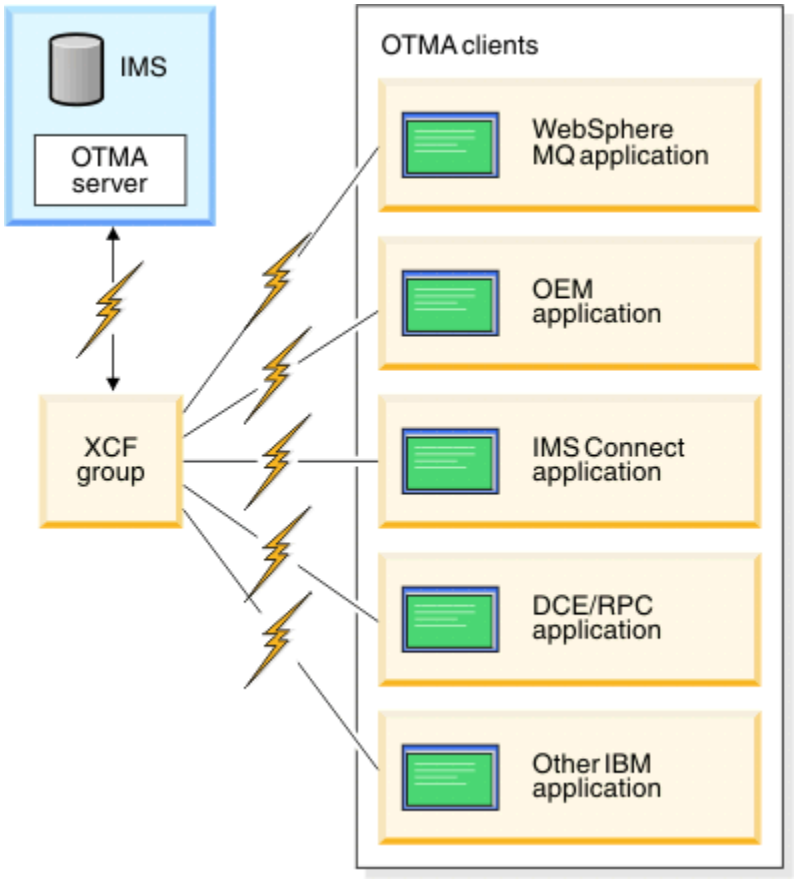


Figure 144. Applications that use XCF to connect to IMS on z/OS

An OTMA client is the gateway by which transactions from outside IMS can enter IMS.

OTMA processing involves:

© Copyright IBM Corp. 1974, 2020

1. A client sends a transaction or command to the server (IMS).
2. The server returns output to the client.

OTMA naming conventions

The names of OTMA clients and transaction pipes must follow a convention defined by IMS.

Names for OTMA clients and transaction pipes must adhere to the following conventions:

- Must be character type A (A-Z, 0-9, @, $)
- Must begin with a non-blank character
- Must be padded with blanks if shorter than the maximum length (16 for a client name, 8 for a transaction-pipe name)
- Cannot contain embedded blanks
- Cannot be a reserved word (for example, "TO" or "SECURITY")
- Cannot begin with "DFS" or "DBCDM"
- Cannot be an IMS keyword (for example, "LINE" or "NODE")

In addition, transaction-pipe names cannot:

- Duplicate an IMS transaction name.
- Have the same name as the z/OS system console (for example, "WTOR"), IMS MTO, or secondary MTO.

IMS does not perform uppercase translation. If lowercase characters are used, the client receives a negative acknowledgment (NAK) response from the server.

Messages sent by OTMA clients

An OTMA client communicates with IMS by sending messages.

First, a user enters application data using a device or program that is connected to the client. Next, the client adds some information (the message prefix) and sends the message to IMS. Output from IMS is sent to the client as a message, and the client uses the message prefix to route the data to the correct device or program.

Parts of the OTMA message prefix

The OTMA message prefix has the following sections:

- Message-control information

  This section includes the transaction-pipe name, message type, sequence numbers (if any), the time out value for send-then-commit messages, and various flags and indicators.
- State data

  This section includes a destination override (if any), map name, synchronization level, commit mode, tokens, and server state.
- Security data

  This section includes the user ID, user token, and security flags.
- User data

  This section includes any special information needed by the client.

Following the message prefix is the application-data section of the message. This section contains either the data to be sent to IMS for processing or the IMS response.

OTMA message-prefix rules

Because a message can have a single segment or multiple segments, the following rules apply to OTMA message prefixes:

- Single-segment messages can have the full prefix (message-control information, state data, security data, and user data).
- Only the first segment of multi-segment messages has the full prefix. Subsequent segments are sent with only the message-control information and application-data sections.
- Acknowledgment (ACK or NAK) messages sent by IMS only return the first input buffer. This message carries the full prefix, and the application-data section (if it is included in the client request).

Sequence numbers used by OTMA

OTMA uses two types of sequence numbers for messages: send-sequence numbers and recoverable sequence numbers. Send-sequence numbers and recoverable sequence numbers are used differently in OTMA.

Using send-sequence numbers

Send-sequence numbers are used for input and output messages. Send-sequence numbers should be incremented by a client for every input message. When IMS sends output to a client, the send-sequence numbers in the output message are also incremented. The send-sequence numbers are used for all the OTMA input/output messages. The send-sequence numbers in the input messages are also used to identify multi-segments.

For example, there is a two-segment OTMA input message. The first segment message will have `send-sequence number`=XXX and `segment number`=1. The second segment message should have the same `send-sequence number`=XXX and `segment number`=2. OTMA chains the two-segment message together because the send sequence numbers are the same.

OTMA uses send-sequence numbers in the following ways:

- All ACK and NAK messages from IMS use the send-sequence numbers submitted by the client on input.
- All OTMA commands that IMS sends to the client have send-sequence number 0 (zero). And, except for the resynchronization flows, these OTMA commands are all single segment.
- Send-sequence numbers for IMS error messages and IMS transaction output are set for each transaction pipe. The send-sequence number for a given transaction pipe is incremented by one for each message, and it is never 0 (zero). When the sequence number exceeds 4,294,967,295 (the 32-bit maximum), it is reset to 1.

Using recoverable sequence numbers

Recoverable sequence numbers are used only to control resynchronization. If a client does not support resynchronization, `recoverable sequence number=0 (zero)`. Resynchronization is only valid for synchronized `tpipe` and commit-then-send input/output. The recoverable sequence numbers are also incremented for every input/output message. Resynch support has an added logic to check if the recoverable sequence numbers are properly incremented. If the sequence numbers are not properly incremented, a NAK is sent. Because the resynch is dependent on the recoverable sequence numbers, the resynch must be correct for every input/output. Recoverable sequence numbers apply to transaction pipes, which use them to control resynchronization.

Related concepts

“Client/server resynchronization with OTMA” on page 837

In order to guarantee that client transactions are processed and that they are processed only once, OTMA provides a protocol for synchronizing transactions.

Related reference

"OTMA message prefix" on page 859
OTMA messages have a prefix that conforms to a format that is mapped by the DFSYMSG macro in the IMS.ADFSMAC data set.

Sending type-1 commands from an OTMA client

You can submit type-1 IMS commands from an OTMA client application. Some restrictions and recommendations apply.

Most commands should be submitted with the send-then-commit (CM1) protocol. However, the following commands require the commit-then-send (CM0) protocol:

- **/DBDUMP DATABASE**
- **/DBRECOVERY AREA|DATABASE**
- **/START AREA|DATABASE**
- **/START REGION**
- **/STOP AREA|DATABASE**
- **/STOP REGION**

The **DISPLAY TRANSACTION** command behaves differently when it is issued from an OTMA client. The command sends its output directly to the client, not to the IMS master terminal. Depending on the setting of the extended-response-requested flag in the message-control information section of the message prefix, the output is either in an architected format (only supported for this command) or in the standard IMS format.

Recommendation: Because the client must use the commit-then-send (CM0) flow, the output from these commands cannot be tied to the input command. The OTMA prefixes are not replicated (the only field common to both the input and the output is the transaction-pipe name). Therefore, configure the client to submit IMS commands using a transaction pipe that the client reserves for IMS command processing.

Restrictions:

OTMA clients cannot submit IMS commands from the following types of subsystems:

- An IMS Extended Recovery Facility (XRF) alternate subsystem
- A CICS-IMS DBCTL subsystem

Related reference

/DISPLAY TRAN command (Commands)

OTMA commit processing

OTMA can control how IMS commits transactions: they can be either commit-then-send or send-then-commit.

Definitions:

- For *commit-then-send* transactions (the IMS standard flow), IMS processes the transaction and commits the data before sending a response to the OTMA client.
- For *send-then-commit* transactions, IMS processes the transaction and sends a response to the OTMA client before committing the data.

Q: What is the major difference between the commit-then-send processing option and the send-then-commit processing option?

A: The commit-then-send processing option commits the transaction output as part of sync-point processing, and then delivers the output to the client later.

The send-then-commit processing option delivers the transaction output first, receives an acknowledgment from the client, and then completes the sync-point processing.

Q: What happened to the commit mode 0 and commit mode 1 processing options?

A: Commit mode 0 is now called "commit-then-send", and commit mode 1 is called "send-then-commit". Because the terms "commit-then-send" and "send-then-commit" are more intuitive when referring to these processing options, the terms "commit mode 0" and "commit mode 1" are no longer used.

For an OTMA transaction, a client can receive one of the following from IMS:

- An ACK message for the input, followed by any output messages.

  In addition send-then-commit transactions will also receive an ACK message followed by a "deallocate" flow (indicated when the commit-confirmation flag in the message-control information section of the message prefix is set to either Committed or Aborted).
- A NAK message with a sense code.
- A NAK message with the processing flag set to `Error Message Follows` in the message-control information section of the message prefix. The subsequent message has the same message prefix as the NAK message and has the IMS error message in the application-data section of the message prefix.

Summary of OTMA commit processing

The following table summarizes the differences between commit-then-send and send-then-commit processing.

Several variables are listed in the first column; the differences between processing options are described in the next two columns. Following the table are some usage notes to be aware of.

Table 146. Commit-then-send versus send-then-commit processing

Variables	Commit-then-send	Send-then-commit
Conversational	Client receives a NAK message.	Supported.
Fast Path	Client receives a NAK message.	Supported.
Non-conversational and non-Fast Path transactions	IMS commits after enqueuing the output to the client. The output is delivered later.	IMS sends output to the client and then commits.
Enqueue the input?	Yes.	Yes.
Enqueue the output?	Yes.	No.
Synchronized transaction pipe specified?	Supported.	Client receives a NAK message.
Timeout interval enforced?	No.	Yes, if a timeout value is specified with either synclevel=confirm or synclevel=synchpt.

Notes:

- IMS conversations cannot use the commit-then-send commit mode.
- Send-then-commit input and output is irrecoverable.
- For irrecoverable output (send-then-commit), IMS requests an acknowledgment if the synchronization level is set to Confirm.
- For a recoverable transaction, IMS always requests an acknowledgment for an output message.
- For commit-then-send transactions, IMS always requests an acknowledgment.
- Synchronized transaction pipes can only be used for commit-then-send transactions.

- When a send-then-commit (CM1) input message is sent to a transaction, OTMA treats that transaction as RESPONSE mode even if the transaction is defined as NONRESPONSE. If the application does reply to the IOPCB, the output is send-then-commit. If the application does not reply to the IOPCB and does not complete a program-to-program message switch, then OTMA responds with a DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY message.
- When a commit-then-send (CM0) input message is sent to a transaction, and the TMAMHRSP flag is set in the OTMA state data prefix, OTMA treats that transaction as RESPONSE mode even if the transaction is defined as NONRESPONSE. If the application does not reply to the IOPCB and does not complete a program-to-program message switch, OTMA responds with a DFS2082 message.

  Restriction: This DFS2082 message for a commit-then-send transaction occurs only for the original input transaction and would not support the program-to-program switch.

Related information

DFS2082 (Messages and Codes)

Sample OTMA commit processing flows

To show the differences between the two commit modes, the topics listed below show sample flows of data between IMS and clients for each commit mode.

Related reference

"Sample OTMA message flows" on page 831

The following topics show some sample message flows, and describe how various fields in the message prefix are set.

Commit-then-send flow

The commit-then-send flow, also known as the IMS standard flow, enqueues IMS output before sending it to the client. Use this flow for standard transaction processing.

To use the standard flow, specify commit-then-send (commit-mode-0) in the state-data section of the message prefix. This sample flow assumes the following:

- The transaction pipe is synchronized. IMS maintains sequence numbers for recoverable input and output for the transaction pipe.
- Acknowledgment is always requested (by both IMS and the client).

If NAK response is received by IMS, then the output is returned to the queue and will be delivered later.

The flow is illustrated in the following figure. Following the figure is a sequential list that provides more details on the flow.

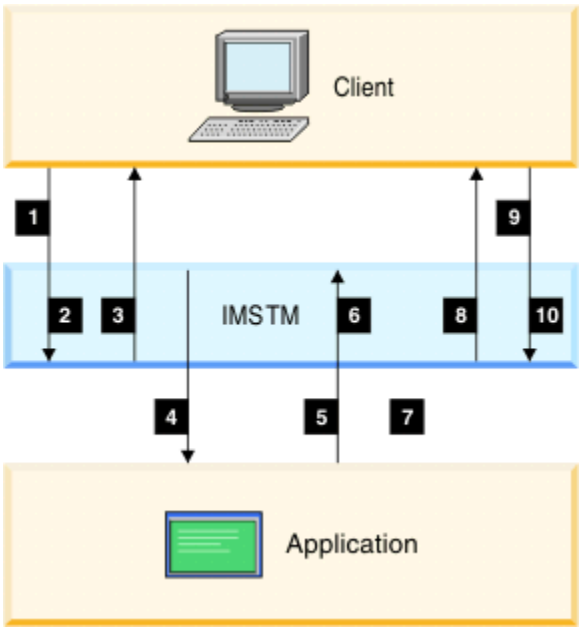


Figure 145. Commit-then-send (IMS standard) flow

The sequence of the flow illustrated in the figure is:

1. Transaction initiated (response required/synchronized tpipe)
2. Transaction is inserted to SMB
3. ACK
4. GU call followed by ISRT to IOPCB
5. Sync Point
6. Output is enqueued to tpipe, and DB is committed
7. Transaction completes
8. Output is sent with response requested
9. ACK
10. Output is dequeued.

An example of the flow of the message activity for a single commit-then-send transaction pipe is shown in the following figure. Following the figure is a sequential list that provides more details on the flow.

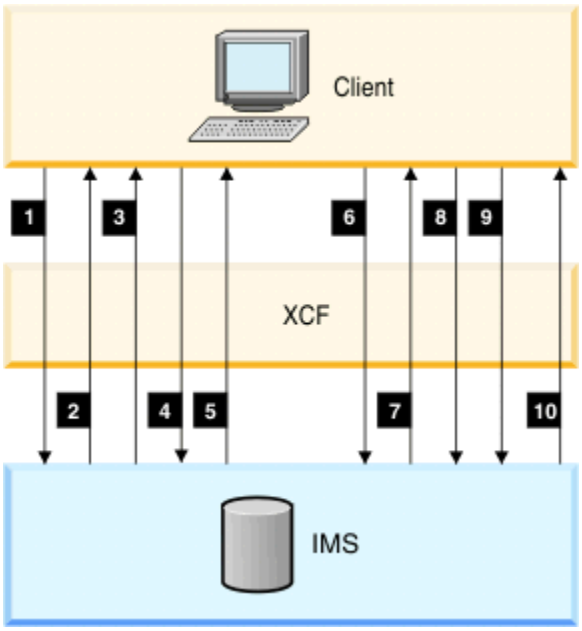


Figure 146. Sample message flow for commit-then-send flow

The sequence of flow shown in the figure is:

1. Tran1
2. ACK to Tran1
3. Tran2
4. Output of Tran1
5. ACK to Tran2
6. Tran3
7. ACK to Tran3
8. Tran4
9. ACK to output of Tran1
10. ACK to Tran4.

OTMA sends a DFS2082 message to the client, regardless of the transaction response mode, when the following conditions are met:

- The TMAMHRSP optional flag is set in the state data prefix.
- The IMS application that processes the original input transaction does not respond to the IOPCB.
- The IMS application that processes the original input transaction does not complete a program-to-program message switch.

Related reference

“Transaction and callout messages” on page 874
The state data for the transaction-related and synchronous callout request information in the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

Related information

DFS2082 (Messages and Codes)

Send-then-commit flow

The send-then-commit (commit-mode 1 or CM1) flow sends IMS output before IMS completes synchronization-point (hereafter referred to as sync-point) processing.

To use the send-then-commit flow, specify Commit Mode 1 in the state-data section of the message prefix. This sample flow assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as None in the state-data section. Therefore, IMS does not request a response (an ACK) when sending output.

The flow is illustrated in the following figure. Following the figure is a sequential list that provides more details on the flow.

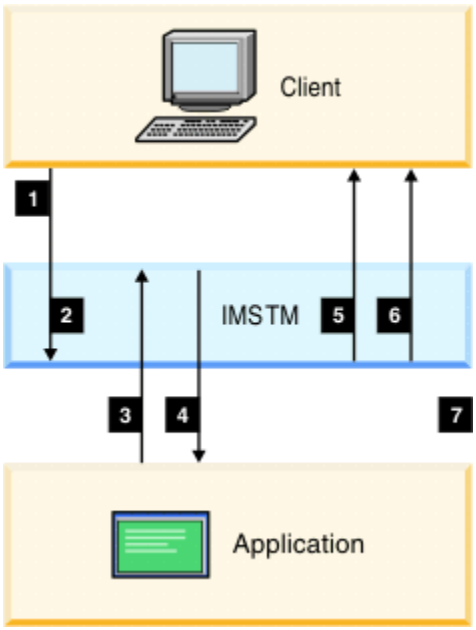


Figure 147. Send-then-commit flow

The sequence of flow shown in the above figure is:

1. Transaction initiated
2. Transaction inserted to SMB
3. GU call followed by ISRT to IOPCB
4. Sync point started
5. Output is sent. No response is requested; response is requested only when sync=confirm is specified.
6. Commit confirmed; IMS completed sync point
7. Transaction completes

An example of the flow of the message activity for a single transaction pipe is illustrated in the following figure. Following the figure is a sequential list that provides more details on the flow.

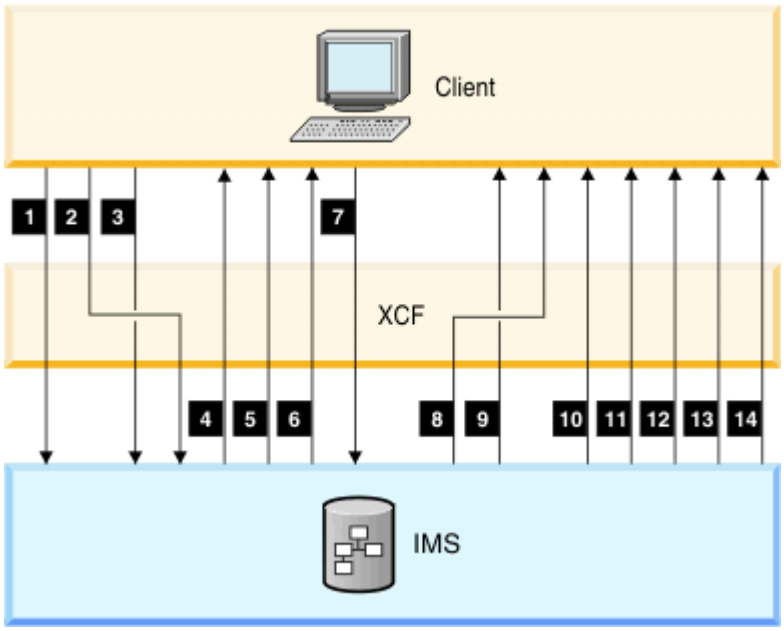


Figure 148. Sample message flow for send-then-commit flow

The sequence of flow shown in the above figure is:

1. Tran1
2. Tran2 request/response
3. Tran3 request/response
4. ACK to Tran3
5. Output of Tran1
6. ACK to Tran2
7. Tran4
8. Output of Tran4
9. Confirm of Tran4
10. Output of Tran2
11. Output of Tran3
12. Confirm of Tran1
13. Confirm of Tran3
14. Confirm of Tran2

As shown in the preceding figure, the client can receive a confirmation for output before receiving the actual output, because z/OS cross-system coupling facility (XCF) does not guarantee that all messages are sent in sequential order. The client must be able to handle this situation during message-receipt processing or by using the XCF Message exit routine.

If a send-then-commit transaction does not run successfully, OTMA sends message DFS2082 to the client and does not perform an insert to the IOPCB.

Send-then-commit flow with confirm

The send-then-commit flow assumes no synchronization for the transactions as they are processed by IMS.

This topic shows a flow in which all transactions are confirmed as they are received (each message requests a response). The sample illustrated in the figure below assumes the following:

- Commit Mode 1 is specified in the state-data section of the message prefix.

- The transaction pipe is not synchronized.
- The Synchronization Level is specified as Confirm in the state-data section.

If NAK is received by IMS, then a user 119ABEND occurs in the application and IMS issues a DFS554 message to the client.

Following the figure below is a sequential list that provides more details on the flow.

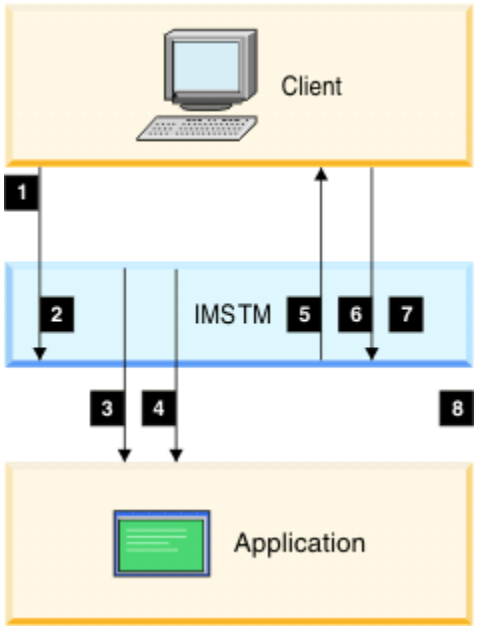


Figure 149. Send-then-commit with confirm flow

The sequence of flow shown in the above figure is:

1. Transaction initiated
2. Transaction inserted to SMB
3. GU call followed by ISRT to IOPCB
4. Sync point start
5. Output sent; response requested
6. ACK
7. DB is committed; commit is confirmed; IMS completed sync point
8. Transaction completed

Sample OTMA message flows

The following topics show some sample message flows, and describe how various fields in the message prefix are set.

In the figures, the following abbreviations are used for parts of the message prefix:

MC
Message-control information section

SD
State-data section

SE
Security-data section

US
User-data section

AP
Application-data section

The sample flow diagrams show which parts of the prefix are mandatory for a given message and which are not applicable. Optional fields and prefix sections are enclosed in parentheses.

For transactions submitted by clients, the following principles apply:

- After IMS sends an ACK message to a client, IMS sends a commit confirmation (indicating that the transaction committed successfully or was aborted).
- The commit confirmation terminates a client transaction.

Related reference

“Sample OTMA messages” on page 887
The following three sample OTMA messages are intended to show what OTMA messages look like when fully constructed, including the parts of the message prefix. The examples are not necessarily related to each other.

Client-bid message flow

The following figure shows a client-bid flow, where the client attempts to connect to the server.

This flow can occur when the client has already joined the z/OS cross-system coupling facility group and notices that a server has joined the group. The client-bid flow is:

1. Client-Bid: MC, SD, SE
2. ACK: MC, SD, SE

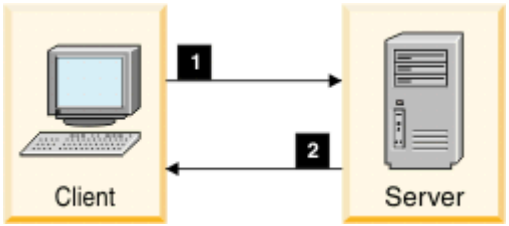


Figure 150. Client-bid flow

The following table shows the contents of the OTMA message prefix in the first flow of a client-bid exchange: the client-bid request. The table shows the contents of the message control data, state data, and security data sections of the OTMA message prefix.

Table 147. Contents of the OTMA message prefix in the first client-bid flow of a client bid exchange

Message prefix section	Contents of prefix section
Message control data	Architecture level = 1 Message type = command Response flag = response requested Command type = client-bid Prefix flag = state data + security data
State data	The state data format for command messages applies to these fields: Length Member name Originator's token Destination token (DRU exit name) MaxBlocksize Aging value Hash table size
Security data	(Utoken)

The following table shows the contents of the OTMA message prefix in the second flow of a client-bid exchange: the acknowledgement. The table shows the contents of the message control data, state data, and security data sections of the OTMA message prefix.

Table 148. Contents of the OTMA message prefix in the second, acknowledgement flow of a client-bid exchange

Message prefix section	Contents of prefix section
Message control data	Architecture level = 1 Message type = command and response Response flag = ACK Command type = client-bid Prefix flag = state data + security data
State data	The state data format for command messages applies to these fields: Length Member name Originator's token Destination token (DRU exit name) MaxBlocksize Aging value Hash table size
Security data	Utoken

Server-available flow

The following figure shows the message flow of a Server-Available exchange, where the server attempts to connect to the client.

This flow only occurs when the server is already joined to the z/OS cross-system coupling facility (XCF) group and recognizes that a client joins the group. A client should not wait for the server to recognize that it has joined the XCF group; the client should send its client-bid message as soon as it joins the group.

The client should ignore a Server-Available message after it has successfully completed its client-bid request and connected to the XCF group. The flow shown is:

1. Server-available: MC, SD, SE
2. Client-Bid: MC, SD, SE
3. ACK: MC, SD, SE

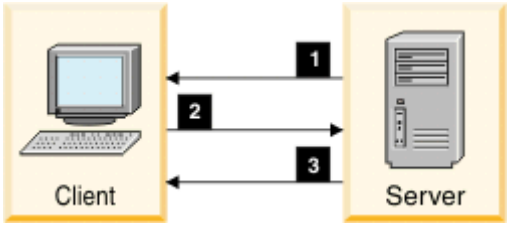


Figure 151. Server-available flow

The following table shows the contents of the message prefix. The flow step is listed, with the message flow type, message prefix section, and associated contents of the message prefix section for the prefixes MC, SD, and SE. The numbers used to show sequence in both the preceding figure and following table are not part of the actual message prefix.

Table 149. Contents of server-available flow message prefix

Flow step	Message flow	Message prefix section	Content of prefix section
1	Server Available	MC	Architecture level = 1 Message type = command No response flag Command type = Server Available
		SD	The state data format for command messages applies to these fields: Length Member name Originator's token Destination token

Table 149. Contents of server-available flow message prefix (continued)

Flow step	Message flow	Message prefix section	Content of prefix section
2	Client-bid	MC	Architecture level = 1 Message type = command Response flag = response requested Command type = client-bid Prefix flag = state data + security data
		SD	The state data format for command messages applies to these fields: Length Member name Originator's token Destination token (DRU exit name) MaxBlocksize Aging value Hash table size
		SE	(Utoken)
3	ACK	MC	Architecture level = 1 Message type = command and response Response flag = ACK Command type = client-bid Prefix flag = state data + security data
		SD	The state data format for command messages applies to these fields: Length Member name Originator's token Destination token (DRU exit name) MaxBlocksize Aging value Hash table size
		SE	Utoken

Commit-then-send transaction flow

The following figure shows the flow for a commit-then-send transaction, where the client submits a transaction to the server for processing.

The flow is:

1. Transaction ABC: MD, SD, SE, (US), AP
2. ACK: MC, SD, SE, (US)
3. Transaction output: MC, SD, (US), AP
4. ACK: MC, SD

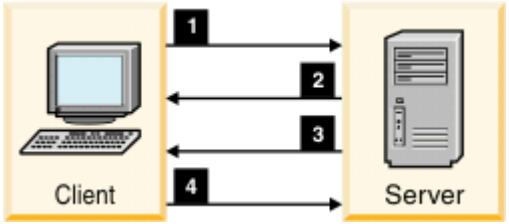


Figure 152. Commit-then-send transaction flow

The following table shows the contents of the message prefix. The flow step is listed, with the message flow type, message prefix section, and associated contents of the message prefix section for the prefixes MC, SD, SE, US, and AP. The numbers used to show sequence in both the preceding figure and following table are not part of the actual message prefix.

Table 150. Contents of commit-then-send transaction flow message prefix

Flow step	Message flow	Message prefix section	Content of prefix section
1	Transaction 'ABC'	MC	Architecture level = 1 Message type = transaction Response flag = response requested Transaction-pipe name Prefix flag = SD/SE/(US)/AP Send-sequence number
		SD	Length Synchronization flag = Commit Mode 0 Synchronization level = Confirm or None (Map name) (Correlator) Length of server user data = 0
		SE	Length (Security flag) Length of fields User ID length (User ID type = 02) (User ID) Profile length (Profile type = 03) (RACF group) Utoken length (Utoken type = 00) (Utoken)
		(US)	This optional section is returned with the transaction output: Length User data
		AP	Length ZZ application data ('ABC' in the example)
2	ACK	MC	Architecture level = 1 Message type = transaction and response Response flag = ACK Transaction-pipe name Prefix flag = SD/SE Send-sequence number
		SD	Length Synchronization flag = Commit Mode 0 Synchronization level = Confirm or None (Map name) (Correlator) Length of server user data = 0
		SE	Length (Security flag) Length of fields User ID length (User ID type = 02) (User ID) Profile length (Profile type = 03) (RACF group) Utoken length (Utoken type = 00) (Utoken)
		(US)	This optional section is returned with the transaction output: Length User data
3	Transaction Output	MC	Architecture level = 1 Message type = data Response flag = response requested Transaction-pipe name Prefix flag = SD/(US)/AP Send-sequence number Server token
		SD	Length Synchronization flag = Commit Mode 0 Synchronization level = Confirm or None (Map name) Server token (Correlator) Length of server user Data (Server user data)
		(US)	This optional section is returned with the transaction output: Length (User data)
		AP	Length ZZ Transaction output data
4	ACK	MC	Architecture level = 1 Message type = data and response Response flag = ACK Transaction-pipe name Prefix flag = SD Send-sequence number
		SD	Length Synchronization flag = Commit Mode 0 Synchronization level = Confirm or None (Map name) Server token (Correlator) Length of server user data (Server user data)

Protecting transactions with OTMA

In a z/OS environment, transaction protection and recovery is managed by z/OS Resource Recovery Services (RRS), part of z/OS Recovery Resource Management Services (RRMS). RRS can apply coordinated changes across multiple mission-critical resources.

OTMA is one of two components that enable IMS to support protected transactions. The second component is APPC/IMS.

To process protected transactions, specify RRS=Y on the control region JCL. IMS then registers as a Resource Manager (RM) with RRMS using the CRGGRM service; it also sets its exits with the Context Services and RRS exit managers using the CRGSEIF service. IMS supports the following RRS exit routines:

- PREPARE
- COMMIT
- BACKOUT
- EXIT_FAILED
- ONLY_AGENT
- SUBORDINATE_FAILED

During initialization, IMS issues message DFS0653I to indicate that it has successfully connected with RRS and that it can now process protected transactions.

Initiating protected transactions from an OTMA client

Perform the following steps to initiate protected transactions from an OTMA client, such as IMS Connect and Db2 for z/OS stored procedures.

1. Specify Synclevel=2 (Syncpt) in the OTMA message prefix.
2. Obtain or reuse a context token. To obtain an RSS context token, use the CTXBEGC service.
3. Set the context token in the OTMA message prefix.
4. Express interest in the unit of recovery (UR) using ATREINT.
5. Send the message to the OTMA.
6. Wait for the output from the IMS transaction.
7. Send an acknowledgment (ACK) to IMS after the output is received.
8. Initiate thez/OS Resource Recovery Services (RRS) commit (ATRACMT) or backout (ATRABCK).

The OTMA client assumes the server distributed syncpoint RM (SDSRM) role. This means that the OTMA client owns the context and is the only RM allowed to initiate or invoke the RRS commit. The flow is similar to that of an APPC/IMS-protected transaction, with the following exceptions:

- The OTMA client initiates the commit using the RRS Commit_Agent_UR service (ATRACMT).
- RRS then directly informs IMS to take a commit. As a result of the previous ATRACMT call, RRS drives the commit exits of all interested RMs.

Processing protected transactions in IMS

IMS receives the protected transaction and extracts the context token from the OTMA message header. IMS saves the context token in its own control block before placing the protected transaction on the IMS message queue.

Also, before placing the protected transaction on the message queue, IMS expresses interest in the context using the Express_Context_Interest service (CTXEINT). IMS will therefore be informed if anything happens to the context while the protected transaction is queued on the message queue.

When IMS schedules the protected transaction into a dependent region, IMS switches the context token to the dependent region TCB using the Switch_Context service (CTXSWCH); it also expresses protected interest in the UR using ATREINT. IMS then presents the protected transaction to an application program that processes that particular transaction. The application contains the business logic (for example,

update databases, send messages, and others). After the application completes its work, it reaches a commit point. IMS then sends the application output back to the OTMA client and waits for a commit or backout event from z/OS Resource Recovery Services.

Client/server resynchronization with OTMA

In order to guarantee that client transactions are processed and that they are processed only once, OTMA provides a protocol for synchronizing transactions.

Using a synchronized commit-then-send (Commit Mode 0) transaction pipe, the client and IMS can regain message flow in the event of a client or IMS outage. Resynchronization occurs when either IMS or the client terminates normally or abnormally.

Transaction resynchronization achieves the following:

- Prevents data from being reprocessed
- Detects that data has not been received and causes the client to resend the data
- Detects that resynchronization might not be possible
- Allows the client to decide what actions to take in order to resynchronize

OTMA resynchronization is not symmetrical, and a system's behavior depends on its role: client or IMS. Resynchronization also does not maintain symmetry for send- or receive-sequence numbers. For example, the differences for the input and output sides of an IMS flow are:

Input
IMS logs the client sequence numbers when the transaction is enqueued, and from that moment, the client has no control over dequeuing the transaction.

Output
The application output is enqueued to a synchronized transaction pipe, but the output sequence numbers are not logged at that time. Only after sending the output and receiving an acknowledgment from the client does IMS finally dequeue the message and log the incremented sequence numbers.

All output using a synchronized transaction pipe is sequenced. The second output message is not sent until the ACK message from the client is received for the first output message.

Q: Why would I use a synchronized tpipe versus a nonsynchronized tpipe?

A: Use a synchronized tpipe to ensure that client transactions are processed only once in the case of a client or IMS failure. Therefore, synchronized tpipes ensure better transaction recoverability. However, in order to guarantee transaction recovery, you are required to implement resynchronization logic with synchronized tpipes.

Use a nonsynchronized tpipe when the recoverability of a transaction is less of a concern. For nonsynchronized tpipes, the client does not require the resynchronization logic.

Assumptions for OTMA resynchronization

The OTMA resynchronization process is based on several assumptions.

The assumptions include:

- Neither client nor IMS sends an ACK message until it has logged a transaction message.
- The client decides what resynchronization actions IMS should take.
- Both client and IMS can determine whether a transaction and its output messages are recoverable. The client can determine a transaction's recoverability using the architected form of the **/DISPLAY TRANSACTION** command.
- Recoverable OTMA messages include a value for the recoverable sequence number in the message-control information section of the message prefix. This value is incremented by 1 every time a recoverable message is sent using a tpipe (see “Summary results of IMS transactions and commands” on page 838).
- A 0 (zero) is not a valid recoverable sequence number.

- Recoverable send- and receive-sequence numbers are maintained on a per transaction pipe basis.
- IMS does not support resynchronization for any IMS command input. If the client needs to submit IMS commands using a synchronized transaction pipe, the recoverable sequence number must be set to 0 (zero). If the recoverable sequence number is not set to 0 (zero), IMS rejects the command input with sense code X'0023'.

Recoverable OTMA transactions

The recoverability of OTMA-initiated transactions and commands is determined by several factors.

The factors include:

- Is it a recoverable or unrecoverable transaction?
- Is it a recoverable or unrecoverable command?
- Is the recoverable sequence number 0 (zero) or not?
- Is it a synchronized or nonsynchronized transaction pipe?
- Is it Commit Mode 0 (commit-then-send) or Commit Mode 1 (send-then-commit)?

A recoverable IMS transaction submitted using the send-then-commit transaction flow is not rejected. However, send-then-commit transactions are discarded during IMS restart (they are unrecoverable).

Transactions that use synchronized transaction pipes are recoverable. Input messages are not recoverable for send-then-commit transactions, and requesting an ACK message has no effect on whether a transaction is recoverable. You should request ACK messages for proper synchronization of the synchronized transaction pipe.

Also, when a transaction reaches IMS, its recoverability depends on how it is defined to IMS.

Unrecoverable OTMA transactions

For unrecoverable transactions, the client must know that the transaction is unrecoverable, process it, and then forget about it.

For send-the-commit transactions, output is unrecoverable and not resynchronized. Also, send-then-commit transactions must be associated with nonsynchronized transaction pipes.

Summary results of IMS transactions and commands

The following figure summarizes the results of IMS transactions that a client submits under various processing conditions using a synchronized tpipe.

The summarization differentiates recoverable sequence numbers of zero and non-zero, and shows the differences between recoverable and unrecoverable transactions for commit modes 0 and 1 for both the zero and non-zero sequence.

In the tables, CM0 indicates a commit-then-send (commit mode 0) transaction and CM1 indicates a send-then-commit (commit mode 1) transaction.

Table 151. Results of IMS transactions using a synchronized Tpipe

Recoverable sequence number	CM0 recoverable transaction	CM0 unrecoverable transaction	CM1 recoverable transaction	CM1 unrecoverable transaction
0 (zero)	Client receives ACK message. Output is recoverable, and no input/ output recoverable sequence is updated.	Client receives ACK message. Output is not recoverable and no input/ output recoverable sequence number is updated.	Client receives NAK message with sense code X'001C'.	Client receives NAK message with sense code X'001C'.

Table 151. Results of IMS transactions using a synchronized Tpipe (continued)

Recoverable sequence number	CM0 recoverable transaction	CM0 unrecoverable transaction	CM1 recoverable transaction	CM1 unrecoverable transaction
Not 0 (zero)	If the recoverable sequence number is valid, client receives ACK message. If it is not valid, client receives NAK message with sense code X'001F'. Transaction and output are recoverable.	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).	Client receives NAK message with sense code X'001C'.	Client receives NAK message with sense code X'001C'.

The following table summarizes the results of IMS transactions that a client submits under various processing conditions using a nonsynchronized tpipe. The summarization differentiates recoverable sequence numbers of zero and non-zero, and shows the differences between recoverable and unrecoverable transactions for commit modes 0 and 1 for both the zero and non-zero sequence.

Table 152. Results of IMS transactions using a nonsynchronized Tpipe

Recoverable sequence number	CM0 recoverable transaction	CM0 unrecoverable transaction	CM1 recoverable transaction	CM1 unrecoverable transaction
0 (zero)	Client receives ACK message. Transaction and output are recoverable.	Client receives ACK message. Transaction and output are not recoverable.	Client receives ACK message. Transaction and output are not recoverable.	Client receives ACK message. Transaction and output are not recoverable.
Not 0 (zero)	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).

The following table summarizes the results of commands that a client issues under various processing conditions using a synchronized tpipe or a nonsynchronized tpipe. The summarization differentiates recoverable sequence numbers of zero and non-zero, and shows the differences between commit modes 0 and 1 for both synchronized and nonsynchronized tpipes.

Table 153. Results of commands that a client issues

Recoverable sequence number	Synchronized tpipe with CM0	Synchronized tpipe with CM1	Nonsynchronized tpipe with CM0	Nonsynchronized tpipe with CM1
0 (zero)	Client receives ACK message. Command output is recoverable and output recoverable sequence number is updated.	Client receives NAK message with sense code X'001C'.	Client receives ACK message. Output is not recoverable.	Client receives ACK message. Output is not recoverable.

Table 153. Results of commands that a client issues (continued)

Recoverable sequence number	Synchronized tpipe with CM0	Synchronized tpipe with CM1	Nonsynchronized tpipe with CM0	Nonsynchronized tpipe with CM1
Not 0 (zero)	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).	Client receives NAK message with sense code X'001C'.	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).	Client receives NAK message with sense code X'0023'. Client should set recoverable sequence number to 0 (zero).

Related concepts

Integrity tables (Application Programming)

Related tasks

"IMS transaction types and transaction states" on page 421

Transactions are the most common type of data that is sent from a logical unit to IMS.

OTMA resynchronization protocol

OTMA resynchronization is based on a series of command exchanges with each client.

The command exchanges for OTMA resynchronization with a client are:

CBresynch (Client_Bid resynch)
CBresynch is sent by the client to request resynchronization with IMS after both the client and IMS have successfully joined the z/OS cross-system coupling facility (XCF) group.

SRVresynch (Server resynch)
SRVresynch must be initiated from IMS to the client after the client has successfully joined the XCF group and issued CBresynch. SRVresynch contains all synchronized tpipe names of which IMS is aware.

REQresynch (Request resynch)
REQresynch must be issued from IMS to the client for each Synchronized tpipe. REQresynch contains the tpipe name, the IMS recoverable send-sequence number for the tpipe, and the IMS recoverable receive sequence number for the tpipe.

REPresynch (Reply resynch)
REPresynch is issued from the client for each tpipe. REPresynch is a reply to the REQresynch request from IMS.

TBresynch (tpipe_Bid resynch)
Tpipe_Bid resynch is issued by the client to initiate resynchronization with IMS for a particular tpipe.

IMS keeps track of the send and receive numbers in the tpipe structure. The send and receive numbers are updated for each input and output message. When resynch occurs, both the client and IMS share their send and receive numbers to verify that both sides are synchronized. The REQresynch command from IMS releases the send and receive numbers from IMS. The client accepts the numbers and does a comparison to the client's send and receive numbers. If the send and receive numbers are not the same, then the client specifies an action to IMS with the REPresynch command. If both sides have the same send and receive numbers, then the resynch completes successfully. If resynch fails, then the failing tpipe is identified and is not used.

Command message exchange for resynchronization must follow the OTMA resynchronization protocol. Normally, the sequence of events that occurs during resynchronization is:

1. The client issues CBresynch when the client attempts to resynchronize with IMS.
2. IMS sends an ACK to acknowledge receipt of CBresynch. From this point on, IMS quiesces any non-resynch type of input or output for all synchronized tpipes. If IMS receives input while

resynchronization is in progress for a synchronized transaction pipe (tpipe), the input is rejected with sense code X'0025'.

3. IMS builds the SRVresynch command and sends it to the client. The SRVresynch command lists all synchronized tpipe names of which IMS is aware for that client.
4. The client receives the SRVresynch command and issues an ACK or NAK message to IMS.
5. If IMS receives an ACK message, IMS begins the resynchronization process for each tpipe. IMS sends the REQresynch command that contains the tpipe name, the IMS recoverable send-sequence number for the tpipe, and the IMS recoverable receive-sequence number for the tpipe.

   If IMS receives a NAK message from the SRVresynch command, IMS sends the DFS2393 message to the MTO and waits for a client-bid request or a CBresynch command from the client.
6. The client receives the REQresynch request. By comparing the information from the REQresynch request with its own information of the tpipe, the client sends the REPresynch reply to IMS and informs IMS about the tpipe
7. IMS receives the REPresynch reply and takes actions on the tpipe, based on the information from the client. IMS sends an ACK message to the client after it has taken actions dictated by the client. IMS enables the tpipe to handle input and output. If IMS cannot perform what the client has requested, IMS stops the tpipe and sends a NAK message to the client.
8. If more than one tpipe exists, the resynchronization process is repeated in parallel for each tpipe. Other tpipes that are not included in the SRVresynch request can send output in either direction anytime after the client receives the SRVresynch command.

The following figure illustrates the flow of nondeferred resynchronization. Following the figure is a sequential list that provides high-level flow description.

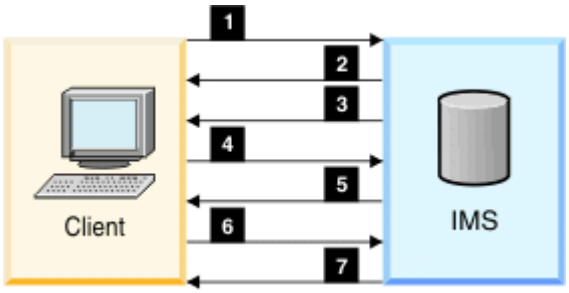


Figure 153. Flow of resynchronization (nondeferred)

1. Client-bid request with resynchronization
2. ACK message
3. SRVresynch command
4. ACK message
5. REQresynch command
6. REPresynch command
7. ACK or NAK message

If the client determines that resynchronization must be deferred for a particular tpipe, the sequence of events for that tpipe differs slightly:

In the REPresynch command, the client can set the "stop and wait for resynchronization" indicator, and can request that IMS defer any input or output while waiting for the TBresynch command from the client. Assuming steps the first four steps have completed, the events following are:

1. IMS sends the REQresynch command that contains the tpipe name, the IMS recoverable send-sequence number and the IMS recoverable receive sequence number.

2. The client receives the REQresynch request. However, due to any product-specific reasons, the client defers resynchronization for this tpipe by sending the REPresynch command with the "stop and wait for TBresynch" indicator on.
3. IMS sends an ACK message to acknowledge receipt of the REPresynch command and waits for TBresynch. Meanwhile, IMS quiesces input and output for the tpipe. If IMS receives any input while waiting for TBresynch, IMS sends a NAK message to the client with sense code X'0025'.
4. The client sends the TBresynch command and requests IMS to resume resynchronization for this tpipe.
5. IMS sends the REQresynch command that contains the tpipe name, the IMS recoverable send-sequence number, and the IMS recoverable receive-sequence number. If the associated tpipe cannot be located using the client's TBresynch command, the client receives a NAK message with sense code X'0025'.
6. The client receives the REQresynch request. By comparing the information from REQresynch request with its own information about the tpipe, the client sends the REPresynch reply to IMS and informs IMS about the tpipe.
7. IMS receives the REPresynch reply and takes actions on the tpipe, based on what the client has requested. IMS sends an ACK message to the client if it has taken actions dictated by the client. Otherwise, IMS sends a NAK message to the client with sense code X'0025' or X'0026'.

The following figure shows the flow of deferred resynchronization. Following the figure is a sequential list that provides high-level flow description.

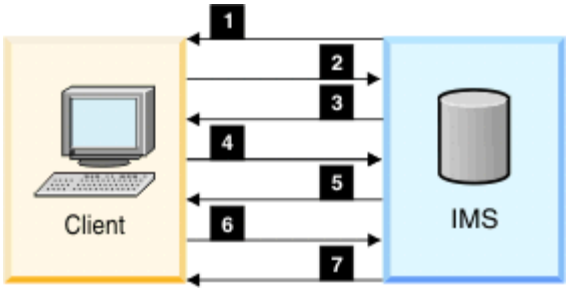


Figure 154. Flow of resynchronization (deferred)

1. REQresynch command
2. REPresynch command with STOP AND WAIT for TBresynch
3. ACK message
4. TBresynch command
5. REQresynch command
6. REPresynch command
7. ACK or NAK message

Related reference

“State data section” on page 868
The state data is mandatory for any OTMA message. It immediately follows the message-control information section in the message prefix. It contains transaction-related information.

Sample OTMA resynchronization message flow

The following figure shows the flow of messages through a synchronized transaction pipe.

Receive- and send-sequence numbers for each side (IMS and client) are represented by the letters **R** and **S**, which are set in the message-control information section of the message prefix. These numbers apply to the entire message (including multi-segment messages). **R** and **S** are not necessarily related.

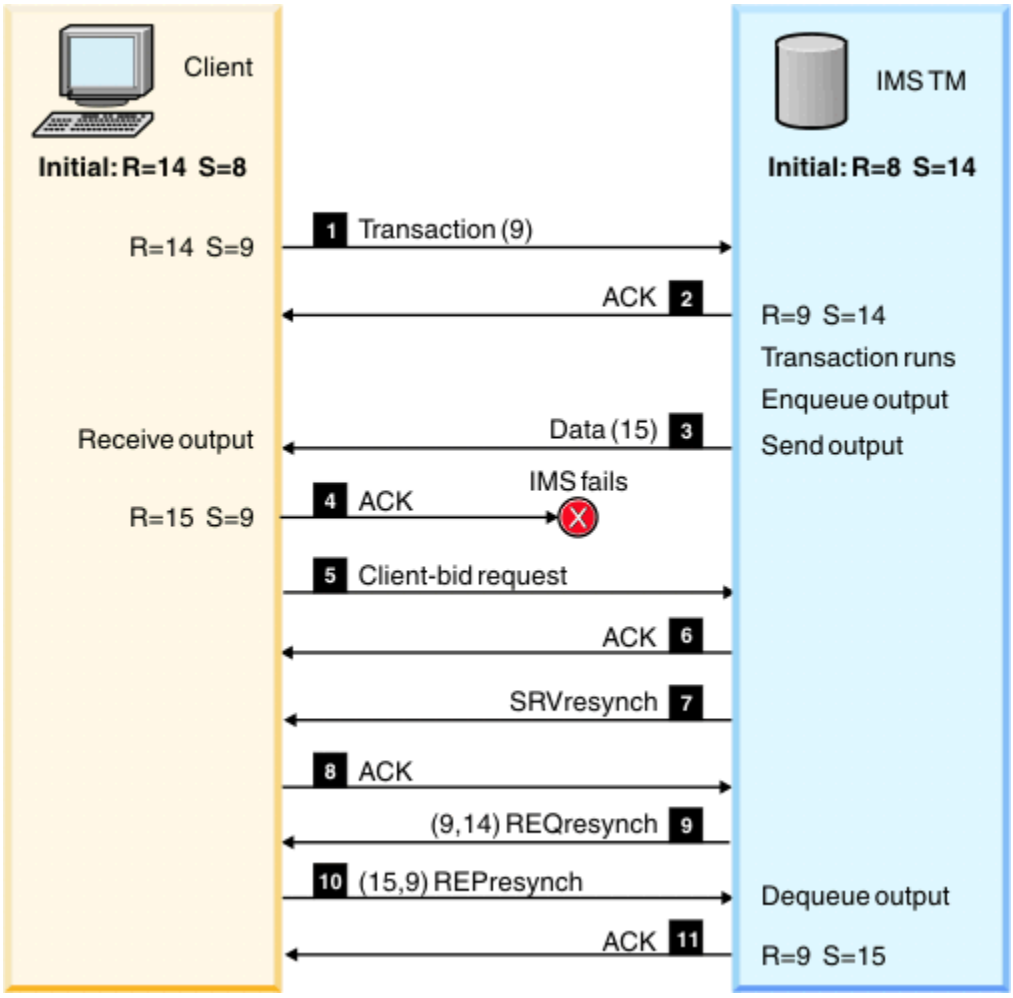


Figure 155. Sample OTMA resynchronization message flow

- 1 After the client submits the transaction, IMS enqueues the transaction, and the transaction runs. The receive-sequence number is incremented by 1.
- 2 IMS sends the client an ACK message to acknowledge receiving and enqueuing the transaction.
- 3 IMS enqueues the output and sends the data to the client.
- 4 The client sends an ACK message to IMS to acknowledge receiving the output; however, IMS never receives this ACK to message 15 because of an IMS failure.

Resynchronization proceeds as follows:

- 5 The client sends a client-bid request to IMS to initiate resynchronization.
- 6 IMS sends an ACK message to the client that resynchronization will begin.
- 7 IMS sends the SRVresynch command to the client to begin resynchronization.
- 8 The client sends an ACK message to IMS to acknowledge receiving the SRVresynch command.
- 9 IMS sends the REQresynch command to the client to update the receive- and send-sequence numbers to (9,14).
- 10 The client sends the REPresynch command to IMS to update the receive- and send-sequence numbers to (15,9), and to tell IMS to dequeue the last output message. IMS dequeues message 15 and updates **S** to 15.
- 11 IMS sends an ACK message to the client.

Sample OTMA resynchronization messages

This topic shows sample OTMA resynchronization messages.

Client-bid request with resynchronization message

The following example shows the OTMA client-bid request with the resynchronization message.

```
MESSAGE CONTROL INFORMATION:

01102000 0C004040 40404040 4040A0C0     |......         .{|
00000000 00000000 00000000 00000000     |................|

STATE DATA:

0036D4D8 E7C3C6F6 40404040 40404040     |..MQXCF6         |
40400100 00010002 00010100 00020002     |  ..............|
0002C4C6 E2E8C4D9 E4F00800 00007FFF     |..DFSYDRU0....".|
FFFF0000 00650056 C3525100 5001A051     |........C...&...|
```

ACK message to acknowledge receipt of CBresynch

The following example shows the ACK message to acknowledge receipt of CBresynch.

```
MESSAGE CONTROL INFORMATION:

01308000 0C004040 40404040 4040A0C0     |......         .{|
00000000 00000000 00000000 00000000     |................|

STATE DATA:

0036D4D8 E7C3C6F6 40404040 40404040     |..MQXCF6         |
40400100 00010002 00010100 00020002     |  ..............|
0002C4C6 E2E8C4D9 E4F00800 00007FFF     |..DFSYDRU0....".|
FFFF0000 00650056 C3525100 5001A051     |........C...&...|
```

SRVresynch command message

The following example shows the SRVresynch command message.

```
MESSAGE CONTROL INFORMATION:

01102000 2C000000 00000000 0000A080     |................|
00000000 00000000 00000000 00010000     |................|

STATE DATA:

000AD1C2 D1F0F0F0 F0C50000 00000000     |..JBJ0000E......|
```

ACK message to acknowledge receipt of SRVresynch

The following example shows the ACK message sent by client to acknowledge receipt of SRVresynch.

```
MESSAGE CONTROL INFORMATION:

0130A000 2C000000 00000000 0000A080     |................|
00000000 00000000 00000000 00010000     |................|

STATE DATA:

000AD1C2 D1F0F0F0 F0C50000 00000000     |..JBJ0000E......|
```

REQresynch command message

The following example shows the REQresynch command message.

```
MESSAGE CONTROL INFORMATION:

01100000 3000D1C2 D1F0F0F0 F0C5A080     |......JBJ0000E..|
00000000 00000000 00000000 00000000     |................|

STATE DATA:

001AD1C2 D1F0F0F0 F0C50000 00020000     |..JBJ0000E......|
00020000 00000000 00000000 00000000     |................|
```

REPresynch command message

The following example shows the REPresynch command message.

```
MESSAGE CONTROL INFORMATION:

01100000 3400D1C2 D1F0F0F0 F0C5A080     |......JBJ0000E..|
00000003 00000000 00000000 00000000     |................|

STATE DATA:

001AD1C2 D1F0F0F0 F0C50000 00020000     |..JBJ0000E......|
00020000 00000000 00000000 00000000     |................|
```

ACK message for successful resynchronization

The following example shows the ACK message sent by IMS to inform the client that resynchronization on a tpipe successfully completed.

```
MESSAGE CONTROL INFORMATION:

01308000 3400D1C2 D1F0F0F0 F0C5A080     |......JBJ0000E..|
00000003 00000000 00000000 00000000     |................|

STATE DATA:

001AD1C2 D1F0F0F0 F0C50000 00020000     |..JBJ0000E......|
00020000 00000000 00000000 00000000     |................|
```

Related reference

"OTMA message prefix" on page 859
OTMA messages have a prefix that conforms to a format that is mapped by the DFSYMSG macro in the IMS.ADFSMAC data set.

Managing commit-then-send output

OTMA provides several different options for managing commit-then-send (CM0) output that is not immediately returned to hold-queue capable OTMA clients. You can purge the output, reroute the output, or share the output among multiple OTMA clients by using an OTMA super member.

Purging commit-then-send asynchronous output

OTMA clients that support asynchronous commit-then-send message output can have OTMA purge output from a tpipe queue when the I/O PCB output cannot be delivered to the OTMA client.

To specify the purge function, the OTMA client must set the purge flag in the OTMA state data prefix of either the original input message or the NAK response when the output cannot be retrieved.

Purging NAKed output

For OTMA clients that are hold-queue capable, OTMA can purge commit-then-send (CM0) I/O PCB output when OTMA receives a NAK response from the client.

If a NAK message turns on the purge request flag in the OTMA state data prefix, OTMA dequeues the commit-then-send I/O PCB output.

If neither a purge request nor a reroute request is specified on a NAK message, OTMA requeues the I/O PCB output to the asynchronous hold queue of the inputting tpipe. Messages stored in the asynchronous hold queue of a tpipe can be retrieved by a RESUME TPIPE call at a later time.

If both a purge request and a reroute request are specified on a NAK message, OTMA requeues the output to the asynchronous hold queue of the inputting tpipe and issues an error message to the system console.

When a purge request is specified on an initial input message, if OTMA cannot deliver the commit-then-send I/O PCB output to the hold-queue capable client due to the z/OS cross-system coupling facility failure or a client outage, OTMA purges the output.

OTMA does not support the purging of asynchronous output from hold queues.

Rerouting commit-then-send asynchronous output

OTMA clients that support OTMA hold queues can have OTMA reroute I/O PCB output to an alternate hold queue when the output cannot be delivered to the OTMA client.

The OTMA client can specify the reroute function on the following types of messages input to OTMA:

- Input messages for commit-then-send transactions
- Input messages for send-only transactions
- NAK response to commit-then-send output

If a maximum number of tpipes has been defined for an OTMA client (tmember) and the number of tpipes associated with the tmember has reached the maximum, a new tpipe cannot be created to reroute a message. In this case, if a message cannot be rerouted because the maximum number of tpipes has been reached, OTMA deletes the message.

The maximum number of tpipes for a tmember is specified on the MAXTP parameter of the OTMA client descriptor in the DFSYDTx PROCLIB member.

Rerouting asynchronous output for send-only transactions

For send-only transactions from OTMA clients that are hold queue capable OTMA can reroute commit-then-send (CM0) messages on the I/O PCB output queue.

To specify a reroute request for asynchronous commit-then-send output in a send-only transaction message, specify the tpipe name of the alternate asynchronous hold queue to which you want the output rerouted in the user data section of the OTMA message prefix. For IMS Connect users, see the OMUSR_REROUT_NM field documented in “OTMA user data fields used by IMS Connect” on page 274.

When reroute is specified, OTMA reroute the output to the reroute tpipe instead of the input tpipe for the send-only message.

OTMA also reroutes to the alternate tpipe any IMS DFS messages generated by the send-only transaction.

However, if a send-only transaction requests an ACK or NAK for the input message, OTMA sends back an ACK after the input transaction is enqueued. If the input send-only transaction cannot be enqueued, OTMA sends back a NAK without a DFS message.

Rerouting NAKed output

For OTMA clients that are hold queue capable, OTMA can reroute commit-then-send (CM0) messages on either the I/O PCB tpipe queue or the hold queue when OTMA receives a NAK response from the client.

If a NAK message turns on the reroute flag and specifies a reroute tpipe name in the reroute TPIPE field of the state data prefix, OTMA reroutes any commit-then-send messages on the I/O PCB output queue by requeuing it to the asynchronous output hold queue specified in the reroute TPIPE field.

If both a purge request and a reroute request are specified on a NAK message, OTMA issues an error message to the system console and requeues the output to the asynchronous hold queue of the tpipe used by the original input message.

When a reroute request is specified on an initial input message, if OTMA cannot deliver the commit-then-send I/O PCB output to the hold-queue capable client due to the z/OS cross-system coupling facility failure or a client outage, OTMA reroutes the output to the asynchronous hold queue of the inputting tpipe.

To request in a NAK message that OTMA reroute asynchronous commit-then-send output, set X'20' in byte 5 in the transaction-related state data of the OTMA header and specify the tpipe name of the alternate asynchronous hold queue to which you want the output rerouted at byte 62 of the state data.

Timeout for acknowledgments of commit-then-send output

You can specify a timeout interval for acknowledgments from the OTMA client, after which OTMA removes commit-then-send messages from the tpipe and places it on a default timeout queue or a queue that you specify.

The timeout value you specify applies to acknowledgments for both commit-then-send output and send-then-commit output; however, upon the expiration of the timeout interval, OTMA reroutes commit-then-send output instead of discarding the output as OTMA does with send-then-commit output.

When an acknowledgment for commit-then-send output expires, OTMA reroutes the output to either a specified reroute tpipe hold queue, a specified timeout tpipe hold queue, or the OTMA default timeout tpipe hold queue DFS$$TOQ.

Related tasks

“Specifying acknowledgment timeout intervals for OTMA messages” on page 801
You can specify an ACK timeout interval that determines how long OTMA waits for an ACK or NAK acknowledgment for OTMA output messages.

Sharing asynchronous commit-then-send output: the OTMA super member function

Hold-queue-capable OTMA clients, such as IMS Connect, can share asynchronous commit-then-send (CM0) output messages by enabling the OTMA super member function. The OTMA super member function is specifically designed to support multiple instances of IMS Connect in a z/OS Sysplex Distributor environment.

The OTMA super member function manages all of the asynchronous CM0 output of all of its participating OTMA clients by using a common output queue. Any participating hold-queue-capable client can then retrieve the CM0 messages on the super member output queue by issuing its own RESUME TPIPE call, regardless of which client the CM0 output was originally destined for.

The messages on the super member queue are retrieved on a first-in-first-out basis, without regard to which instance of the OTMA client originated the output or which instance of the OTMA client issued the RESUME TPIPE call.

The RESUME TPIPE call from a participating OTMA client does not need to specify anything about the super member. The super member function recognizes the RESUME TPIPE call as coming from a participating OTMA client and returns the output on the common queue.

You can register the participation of an instance of IMS Connect in the super member function by specifying the super member name on the SMEMBER parameter of the IMS Connect HWS configuration statement.

To use the OTMA super member in a configuration that includes multiple IMS systems, you must have shared queues enabled for all the IMS systems. If a shared queues back-end IMS creates ALT-PCB output in the super member-enabled environment, the output can be retrieved from any front-end IMS with an OTMA client in the super member set. All retrieval options of the resume tpipe call are supported in the shared queues environment.

If there is only one IMS system with multiple OTMA clients, shared queues support is not required.

To activate the super member function specify a 1- to 4-character super member name in the SMEMBER parameter of the HWS configuration statement in the IMS Connect configuration member (HWSCFGxx). Super member names must be unique and cannot be the same as existing OTMA member names.

The first instance of IMS Connect that specifies a super member name both activates the super member function and defines the name of the super member for all other instances of IMS Connect. Any instances of IMS Connect activated subsequently that are to use the same super member queue must specify that same super member name as specified by the first instance of IMS Connect.

You can display the name of an activated super member by issuing any one of the following IMS Connect commands:

- The IMS type-2 format command **QUERY IMSCON TYPE(CONFIG)**
- The WTOR format command **VIEWHWS**
- The z/OS MODIFY format command **QUERY MEMBER** TYPE(IMSCON)

To display additional information about a super member, you can also issue the IMS commands **/DISPLAY OTMA** and **/DISPLAY TMEMBER TPIPE**.

When you issue the OTMA commands **/START TMEMBER TPIPE**, **/STOP TMEMBER TPIPE**, or **/TRACE TMEMBER TPIPE** with a regular OTMA member specified, OTMA expands the scope of the command to include any related super member.

You can also issue OTMA commands directly to an existing super member. For example, you can dequeue asynchronous messages from a super member by issuing the command **/DEQUEUE TMEMBER** *supermember_name* TPIPE *tpipe_name* PURGE.

Asynchronous output messages that are created by ALT-PCB processing are stored in the super member directly. If the input messages are submitted through an IMS Connect that supports the super member, the input parameter lists for the OTMAYPRX user exit and DFSYDRU0 exit routine contain a flag indicating that the message is from a super member supported client. Also, the OTMA state data prefix pointed to by the input parameter list contains the super member name, if any.

Displaying output on the asynchronous hold queue

You can display the number of messages on the asynchronous hold queue by issuing the /DISPLAY TMEMBER command with the OUTPUT keyword.

The **/DISPLAY TMEMBER TPIPE OUTPUT** command can be used to display the output counts for both primary and hold queues when the OTMA client is hold queue-capable (for example, IMS Connect).

Chapter 45. OTMA support for callout requests

IMS application programs running in IMS dependent regions can call out through the Open Transaction Manager Access (OTMA) component of IMS to servers that are outside of the IMS installation to request data or services. The synchronous callout interface can also be used to request services from another IMS application with a synchronous program switch.

An IMS configuration that supports such callout requests requires multiple components. Which components are required depends on whether the callout requests are processed synchronously or asynchronously.

OTMA is not required for synchronous program switch requests. Synchronous program switch requests are queued as OTMA transactions, but IMS transparently routes the requests to the specified transactions with an internal implementation of the OTMA send-then-commit (CM1) protocol. You do not need to start an XCF connection with an OTMA client, specify OTMA=Y in the DFSPB*xxx* member of the IMS.PROCLIB data set, or issue a START OTMA command.

Callout requests from IMS application programs

OTMA, together with hold-queue-capable OTMA clients such as IMS Connect, supports callout requests from IMS application programs running in IMS dependent regions to data or service providers that are external to the IMS installation.

The external providers can include:

- An Enterprise JavaBeans (EJB) application or a message-driven bean (MDB) application running in a web application server, such as WebSphere Application Server
- A Web service
- A data source or service that is accessed through a user-written IMS Connect client

The request for services or data is referred to as a *callout request*. When an IMS application program makes a callout request, IMS can be viewed as a client in a client-server relationship where the server is the external application to which IMS is making the callout request.

Callout requests that are routed through OTMA can be processed synchronously or asynchronously.

For synchronous callout requests, an IMS application program running in an IMS dependent region issues the DL/I ICAL call and waits in the dependent region to process the response also. When the ICAL call is issued, IMS generates a correlation token for synchronous callout requests and routes the request to an OTMA destination. The correlation token is included with the callout request and must be returned to IMS with the response to route the response back to the requesting IMS application program.

For asynchronous callout requests, an IMS application program running in an IMS dependent region inserts the callout message to an ALTPCB queue and then terminates to free the dependent region. IMS does not generate a correlation token for asynchronous callout requests; consequently, if a response to the callout request is required, the correlation of any must be managed by the IMS application program. When IMS receives a response to an asynchronous callout request, IMS processes the response as a new transaction.

The configuration requirements for synchronous callout requests differ slightly from the requirements for asynchronous callout requests, but the basic components are the same:

- An IMS application that calls out to a source external to IMS for data or services. For synchronous callout requests this same application processes the response. For asynchronous callout requests, if a response is needed, the response is processed either by a different instance of the same application program or a different application program altogether.
- OTMA, which routes the callout request to the appropriate client tpipe queue based on destinations defined by using one of the following methods:
  - OTMA destination descriptor

© Copyright IBM Corp. 1974, 2020

  - Type-2 commands
  - For asynchronous callout requests, the OTMA Destination Resolution user exit (OTMAYPRX) and the OTMA User Data Formatting exit routine (DFSYDRU0)
- A hold-queue-capable OTMA client, such as IMS Connect. IMS Connect serves as both a TCP/IP gateway to IMS and an interface with OTMA.
- If you are using IMS Connect, an IMS Connect TCP/IP client, such as IMS TM Resource Adapter, IMS Enterprise Suite SOAP Gateway, or a user-written IMS Connect client.
- A data or service provider, such as an EJB application or MDB application in a Java Platform, Enterprise Edition (Java EE) environment, or a web service.

Security for both synchronous and asynchronous callout configurations can be implemented by a security product such as RACF, the OTMA Resume TPIPE Security user exit (OTMARTUX), or both.

Related concepts

"OTMA destination descriptors" on page 774
Use OTMA destination descriptors to define destinations for messages that are routed through OTMA.

Synchronous callout requests

Synchronous callout requests are processed in real time and travel from the IMS application program running in an IMS dependent region, out to the external data or service provider, and back to the IMS application program while the IMS application program remains scheduled in the IMS dependent region.

Configuring your environment for synchronous callout can be simpler than configuring your environment for asynchronous callout, because IMS manages the correlation of the callout response to the IMS application that made the request.

Restrictions:

- OTMA does not support synchronous callout requests in shared-queues environments that are configured with front- and back-end IMS systems.
- The OTMA Input/Output Edit user exit (OTMAIOED) is not supported for either synchronous callout request messages received by OTMA from IMS application programs that issue the DL/I ICAL call or synchronous callout response messages.

OTMA processes synchronous callout request messages as nonrecoverable commit-then-send (CM0) output messages. However, IMS processes synchronous program switch requests in send-then-commit mode (CM1). Synchronous program switch requests are not routed to external clients such as IMS Connect, and you do not have to enable OTMA to use them.

To increase the throughput of outgoing callout messages, consider enabling OTMA support for multiple active RESUME TPIPE requests, which enables a single OTMA tpipe to send output to multiple clients in parallel. Support for multiple active RESUME TPIPE requests can be enabled by either an OTMA client descriptor or in the IMS Connect system or data store definition.

The following high-level steps provide an overview of configuring a synchronous callout environment when IMS Connect is used as the OTMA hold-queue capable client:

1. Code an OTMA destination descriptor to route the callout request to the tpipe of the IMS Connect client that is configured to retrieve the callout requests.
2. Restart IMS.
3. If you need to modify an IMS Connect user message exit routine, reassemble and bind the IMS Connect user message exit routine.
4. If you modified an IMS Connect user message exit routine, restart IMS Connect.
5. Configure an IMS Connect TCP/IP client to retrieve the callout request, pass the request to the data or service provider, and return the response to IMS Connect. For information about how to configure your IMS Connect TCP/IP client see the appropriate documentation listed below:
   - Callout programming models (TM Resource Adapter)

- IMS Enterprise Suite SOAP Gateway overview
- Configuring user-written IMS Connect clients for synchronous callout requests

6. Code an IMS application program to initiate callout requests by issuing the DL/I ICAL call and to process the response.

When callout service providers connect to IMS through an OTMA client such as IMS Connect, the synchronous callout responses are sent back to OTMA and IMS using the send-only protocol.

Optionally, a callout service provider can use the *send-only with ACK* protocol, which requires OTMA to issue an ACK or NAK reply to each response from the callout service provider. The send-only with ACK protocol also requires the callout service provider to issue an additional receive to retrieve the ACK or NAK message.

By default, OTMA includes the data from the response message in each ACK message when the send-only with ACK protocol is used. You can omit the response data from each ACK by specifying X'10' in the Synchronization or Callout Flags field at byte 3 of the state data section of the OTMA message prefix.

IMS provides various sample application programs to test callout support. For more information, see Samples for the callout function (Installation).

After you complete the initial setup of the configuration for callout requests, you can use the DL/I test program (DFSDDLT0) to verify and debug ICAL DL/I call independently of your IMS application programs.

Related concepts

"Synchronous program switch requests" on page 851
Make a *synchronous program switch request* by using a DL/I ICAL call to request and receive a response from another IMS transaction in the same unit of work. To make ICAL calls directly from a Java-enabled dependent region, use the IMSCallout API. To learn more, see Programming with the Callout API (Application Programming).

"Send-only protocol for synchronous callout responses" on page 299
IMS Connect clients return responses to synchronous callout requests from IMS application programs by using the send-only protocol.

Related tasks

"Configuring user-written IMS Connect clients for synchronous callout requests" on page 201
To support synchronous callout requests, user-written IMS Connect clients must be configured to retrieve new callout requests from IMS, acknowledge the receipt of the callout request (ACK or NAK), and to return the synchronous callout responses to IMS through IMS Connect.

"Returning callout responses to IMS" on page 207
User-written IMS Connect clients return the callout response messages to IMS by using the send-only protocol for synchronous callout responses.

Synchronous program switch requests

Make a *synchronous program switch request* by using a DL/I ICAL call to request and receive a response from another IMS transaction in the same unit of work. To make ICAL calls directly from a Java-enabled dependent region, use the IMSCallout API. To learn more, see Programming with the Callout API (Application Programming).

The target transaction runs in a separate unit of work, so it is not eligible for two-phase commit or z/OS Resource Recovery Services.

The target transaction can be in the same IMS system, a remote IMS system connected through multiple systems coupling (MSC), or an IMS system that is accessible through shared queues.

IMS schedules the transaction for the ICAL call as an OTMA transaction. You do not need to enable OTMA to make a synchronous program switch request.

To use the synchronous program switch function, you must first configure an OTMA destination descriptor with TYPE=IMSTRAN for the destination application. The destination descriptor can be set in the DFSYDT*x* member of the IMS.PROCLIB data set, or you can create it with the **CREATE OTMADESC** command.

The TMEMBER, TPIPE, and SMEM parameters behave differently for the IMSTRAN descriptor type than for other descriptor types. These parameters are used to optionally specify a default destination for late response messages from the target application program.

The following figure shows the five-step synchronous program switch processing model.

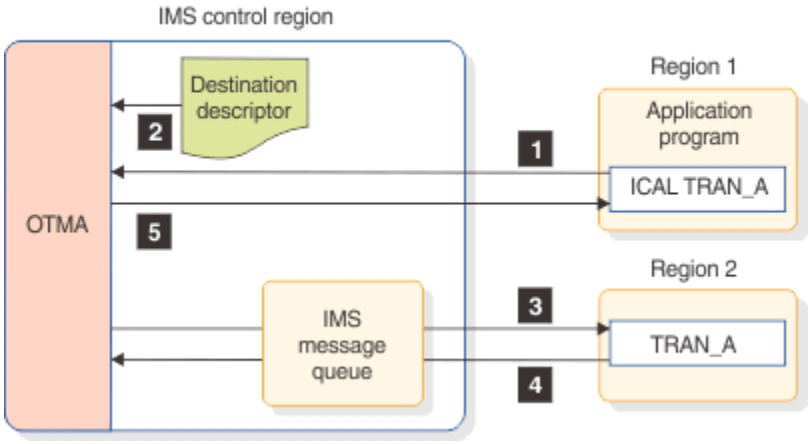


Figure 156. IMS synchronous program switch processing

1. The application program that is running in region 1 issues an ICAL to TRAN_A. The ICAL request is routed to the IMS control region.
2. The IMS control region reads the destination descriptor that is specified in the request.
3. The IMS message queue is used to schedule the TRAN_A transaction in region 2 using the send-then-commit (CM1) protocol. The OTMA routing function is used even if OTMA is not enabled in the IMS system.
4. After the target application finishes processing, it returns the response message to the IMS control region.
5. The IMS control region responds to the original DL/I ICAL call in region 1 with the response message from region 2.

Note: The dependent regions can be MPPs, JMPs, IFPs, BMPs, or JBPs.

IMS uses the OTMA message header format for synchronous program switch messages. In the state data header, the commit mode is set to 1 and the sync level is set to CONFIRM. The TMAMHCOR field is set to the LCRETOKN value from the IMS region where the ICAL call was issued. However, if the synchronous program switch originated from an OTMA transaction, IMS inserts information from the original transaction: TMAMHCOR is set to the original correlator value, TMAMHCID is set to the original client member name, and TMAMRTOD is set to the original tpipe name.

The user ID name from PSTUSID and the group name from PSTGRPNM are included in the security data header.

No user data header is needed if the original transaction is not an OTMA transaction and the OTMA destination descriptor for the message does not include any late message routing information. If the ICAL call is from an OTMA transaction that includes the user data header information, the same user data header information is used for the synchronous program switch message.

Restrictions:

- The OTMA Input/Output Edit exit routine (DFSYIOE0) is not called for a synchronous program switch request or for response message.
- The TM and MSC Message Routing and Control exit routine (DFSMSCE0) is not called for a synchronous program switch request.
- The target transaction is not part of the RRS commit scope of the initiating application program.

- BMP and JBP applications cannot make synchronous program switch requests in a DBCTL environment.
- The target transaction has read-only access to Fast Path MSDBs.
- The target transaction cannot be an IMS conversational transaction.
- To use synchronous program switch in a shared queues environment, all of the participating IMS systems must be Version 13 or later.

Issuing synchronous program switch requests from Java applications

To make ICAL calls directly from a Java-enabled dependent region, use the **IMSCallout API**. To learn more, see Programming with the Callout API (Application Programming).

Synchronous program switch requests can also be issued from Java applications running in IMS Java dependent regions. The IMS Java dependent region resource adapter uses the Java Message Service (JMS) API as a front end for synchronous program switch requests.

Related tasks

Implementing the synchronous callout function (Application Programming)

Issuing synchronous program switch requests from a Java dependent region (Application Programming)

Related reference

OTMA destination descriptor syntax and parameters (System Definition)

ICAL call (Application Programming APIs)

Asynchronous callout request

IMS application programs running in IMS dependent regions can send outbound messages to request services or data from a WebSphere Application Server EJB application or a web service. The request for services or data is referred to as a **`callout request`**.

Asynchronous callout requests do not require the IMS application program that issues the request to wait for a response in the dependent region. Upon issuing an asynchronous callout request, the application program can terminate and free the dependent region. Any response to the callout request that is returned to IMS is treated as a new incoming transaction and IMS schedules a new application program instance to process it.

If an asynchronous callout request generates a response, however, the benefit gained by freeing dependent regions might be offset by the additional complexity of managing the response. For asynchronous callout responses, your installation is responsible for developing the method for correlating the response to the original request. For synchronous callout requests, IMS manages correlation for you.

IMS provides various sample application programs to test callout support. For more information, see Samples for the callout function (Installation).

Implementing the asynchronous callout function

Implementing an IMS Configuration that supports the asynchronous callout function involves planning for correlation, application programming, and OTMA routing.

Planning for correlation of asynchronous callout responses

If a callout request generates a response, the response is treated as a new transaction that runs independently from the original transaction that generated the callout request. Consequently, you must correlate the response to the appropriate IMS transaction.

Planning for correlation involves deciding what data to use for correlation and whether or not to store any correlation data in an IMS database until the callout response returns.

You also need to ensure that the tpipe name used by the IMS application in the ISRT *alternate_pcb* call and the tpipe name used by either the EJB or the IMS Enterprise Suite SOAP Gateway connection bundle for the RESUME TPIPE call both point to the same OTMA asynchronous hold queue either by using the same name for each or by coding the appropriate logic in the OTMA destination routing exit routines. The

tpipe name is based on the alternate client ID provided by the external application through IMS Connect. The application program issuing the callout request must specify the tpipe name of the asynchronous hold queue in an ISRT *alternate_pcb* call.

Application programming for asynchronous callout requests

An application program issuing an asynchronous callout request must include in the callout request message any data required by the external application it is calling.

If the correlation of a response from the external application is required, the calling application program must also capture any data required for correlation purposes. The application program must also include the tpipe name of the dedicated OTMA asynchronous hold queue for the callout request function.

If any correlation data is included in the callout request message, you must ensure that the external application program tolerates the data and sends it back with the response.

Optionally, the application program might also store data in a database for correlation purposes or for the results of any IMS processing that must be returned with the callout response.

An application program processing a response to a callout request can process any correlation data returned by the external application program, retrieve any correlation data or IMS data stored in the callout database, and return the final data to the appropriate queue, terminal, or user as appropriate.

If you are using IMS TM Resource Adapter, a Java application that processes a callout request externally can be configured to "listen" to the OTMA asynchronous hold queue by issuing a looping RESUME TPIPE call with an appropriate wait time. The Java application program might also be designed to preserve the correlation data and send it back with the callout response.

Related concepts

“IMS application programs and the asynchronous callout function” on page 856
You might need to modify or create IMS application programs to support the callout function if you need IMS to process a response to the callout request, a terminal requires an acknowledgment after submitting a transaction, or a response-mode transaction initiates a program-to-program switch.

OTMA routing of asynchronous callout requests

OTMA routing of asynchronous callout requests is required to route the `ISRT alternate_pcb` call to the appropriate tpipe.

The OTMA routing is performed by the OTMA destination descriptor or the OTMA Destination Resolution user exit (OTMAYPRX) and the OTMA User Data Formatting exit routine (DFSYDRU0).

You can allow the exit routines to override a defined destination descriptor by setting EXIT=YES for the descriptor. Otherwise, the destination descriptor is used and the exit routines are not called.

How the OTMA routing of the callout response is handled depends on your installation. The callout response can be routed back to a user or terminal or to an OTMA asynchronous hold queue.

Related concepts

“OTMA destination descriptors” on page 774
Use OTMA destination descriptors to define destinations for messages that are routed through OTMA.

Related reference

OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits) (Exit Routines)
OTMA User Data Formatting exit routine (DFSYDRU0) (Exit Routines)

Initiating asynchronous callout requests from an IMS Connect TCP/IP client

To initiate an asynchronous callout request from an IMS Connect TCP/IP client, the TCP/IP client must submit a Send Only request that invokes an IMS application program that then issues the callout request.

To wait for and retrieve the response to the callout request, the TCP/IP client then must issue a Resume Tpipe call.

IMS TM Resource Adapter and asynchronous callout requests

If you are using the asynchronous callout function with the IMS TM Resource Adapter, your WebSphere Application Server administrator must configure a shareable connection factory to be used by the EJB bean or message-driven bean to retrieve the callout requests from IMS through IMS Connect.

The bean uses the IMS TM Resource Adapter to "listen" to the OTMA asynchronous hold queue by issuing a looping RESUME TPIPE call with a long timeout value. If no messages are on the queue when a RESUME TPIPE call is issued, the IMS TM Resource Adapter remains in a wait state. If the RESUME TPIPE call times out before a callout request arrives on the asynchronous hold queue, control is returned to the bean, which can then issue another RESUME TPIPE call to continue listening or break the loop to perform error checking.

Also, your Java application programmers must modify the bean to specify an alternate client ID that identifies the OTMA tpipe on which the callout request is queued.

Optionally, you can have your bean process the callout request itself, or you can have the bean pass the callout request on to a web service.

If you are using the IMS TM Resource Adapter with a z/OS Sysplex Distributor that distributes input to IMS through multiple instances of IMS Connect, the callout requests must be made using a TMEMBER for which the OTMA super member function is enabled.

For information about coding the IMS TM Resource Adapter and EJB applications, including sample application code, see and Callout programming models (TM Resource Adapter).

SOAP Gateway and asynchronous callout requests

IMS Enterprise Suite SOAP Gateway includes support for the asynchronous callout function, which enables IMS application programs to make asynchronous callout requests to web services through the SOAP Gateway.

When enabled for asynchronous callout requests, the SOAP Gateway listens for new callout requests by continuously issuing the RESUME TPIPE call to IMS Connect. If a callout request message is on the tpipe queue, OTMA sends the callout request to IMS Connect, IMS Connect processes the message, converting the message to XML if necessary, and then sends the message to the SOAP Gateway.

If you are using the SOAP Gateway with a z/OS Sysplex Distributor that distributes input to IMS through multiple instances of IMS Connect, the callout requests must be made using a TMEMBER for which the OTMA super member function is enabled.

If the callout request message requires conversion to XML, XML conversion support must be enabled in IMS Connect and the callout request message must specify the XML adapter and the converter in its OTMA header of the callout request message. If the callout request message does not specify an XML adapter and a converter, IMS Connect does not convert the message to XML.

You can specify an XML adapter and converter for callout request messages by using the OTMA destination descriptor. For example:

```
D DESCNAME TYPE=IMSCON TMEMBER=HWS1 TPIPE=PIPENAME
D DESCNAME ADAPTER=HWSXMLA0 CONVRTR=CNVNAMED
```

In the example, if an IMS application program inserts a message to the ALT IOPCB using the descriptor name DESCNAME, OTMA adds the adapter and converter names to the OTMA header of the callout request message and then queues the message to the tpipe PIPENAME.

IMS Connect support for XML conversion with the asynchronous callout function requires a callout XML converter for each web service that you want your IMS application program to call. Generate the callout XML converters by using Rational® Developer for System z Version 7.1.1 or later.

For more information about the SOAP Gateway, see the SOAP Gateway documentation in the IBM Documentation.

Related concepts

"Overview of IMS Connect XML Conversion Support" on page 154
For certain IMS Connect clients, IMS Connect can convert the XML data contained in an input message into the data structures used by IMS application programs written in either COBOL or PL/I. The data in the corresponding output message is also converted from programming language of the IMS application program back to the XML data that IMS Connect client expects.

"OTMA destination descriptors" on page 774
Use OTMA destination descriptors to define destinations for messages that are routed through OTMA.

IBM MQ and asynchronous callout requests

IBM MQ supports asynchronous callout requests from the ALT IOPCB interface.

OTMA can be configured with destination descriptors for IBM MQ applications. The MQ message descriptor (MQMD) data structure expected by IBM MQ contains fields that control MQ message handling. The OTMA descriptor contains parameters to configure the MQMD structure, which can be set either with the DFSYDT*x* member of the IMS.PROCLIB data set, or with type-2 commands.

A descriptor that matches the destination sets the default routing and message handling parameters. However, you can use the OTMA routing exits (DFSPRX0 and DFSYDRU0) to override the routing information in the destination descriptor if they are enabled with the EXIT=YES parameter for that descriptor.

Related concepts

"OTMA destination descriptors" on page 774
Use OTMA destination descriptors to define destinations for messages that are routed through OTMA.

Related reference

OTMA destination descriptor syntax and parameters (System Definition)

IMS application programs and the asynchronous callout function

You might need to modify or create IMS application programs to support the callout function if you need IMS to process a response to the callout request, a terminal requires an acknowledgment after submitting a transaction, or a response-mode transaction initiates a program-to-program switch.

If you do not need a response to the callout request, the application program needs to specify only the appropriate OTMA destination name in an insert call to an alternate PCB. No modification of the application program is necessary.

If, however, you need a response to the callout request, you must modify the IMS application programs to correlate the response to the original input transaction. The method of correlation is ultimately up to you and your installation, but it will very likely require the application program to include in some type of data to be used for correlation purposes in the outgoing callout request message and then look for that same data in the response message.

You might also have the IMS application program that issues the callout request store correlation data or other data in an IMS database for retrieval when the callout response is returned. In a shared queues environment, such a database is also useful if the response transaction might be scheduled on a different IMS system. If a reply must be sent back to either an LTERM or an IMS Connect TCP/IP client, the LTERM name or TMEMBER and TPIPE name should also be stored in the database.

The following code samples shows an example of a COBOL copybook data structures that might be used for the callout function and the callout response.

The callout request data structure includes a correlator field, CORRID, to send correlation data to the EJB:

```
01 CALLREQ
     02   LL        PIC S9(4) COMP.
     02   ZZ        PIC S9(4) COMP.
     02   CORRID    PIC (8).
     02   ACCTNUM   PIC (20).
```

When the response is returned, the CORRID field of the COBOL copybook data structure contains the same correlation data, which the IMS application can use to relate the response to the initial request, if needed:

```
01 CALLRESP
     02   LL        PIC S9(4) COMP.
     02   ZZ        PIC S9(4) COMP.
     02   CORRID    PIC(8).
     02   ACCTBAL   PIC (20).
```

Avoiding hung application programs and terminals with asynchronous callout requests

Generally, terminals do not implement any timeout mechanisms and if a response is not returned to the terminal, the terminal hangs.

A callout request could result in a hung terminal in at least two possible cases: the application program issuing a callout request terminates without responding to the terminal or a response-mode transaction performs a program-to-program switch without adequate correlation of the response to either the callout request or the inputting terminal.

The first case might occur if, for example, a terminal waits for a response after submitting a transaction that initiates a callout request, and the IMS application program processing that transaction terminates after inserting the callout request to the ALTPCB, the terminal might hang waiting for a response. To avoid this, IMS application programs that issue a callout request should be modified to send an acknowledgment to the inputting terminal.

In the second case, a terminal might hang in the following scenario: an application program, PGMA, processes a response-mode transaction that waits for a response after initiating a program-to-program switch to application program, PGMB, which in turn issues a callout request that generates a response to IMS. Then, if a third application program, PGMC, processes the response, but does not notify PGMA, PGMA hangs waiting for a response. To avoid this, PGMA should be modified to support the correlation of the response through PGMB and PGMC, so that PGMC can pass the response to PGMA for return to the initiating client terminal. If a callout request will not generate a response, response-mode transactions should not be used with applications program such as PGMA.

Correlating responses to asynchronous callout requests

If the external application or web service returns a response message to IMS as a result of a callout request, the IMS application programs processing the request and response will need to correlate the response message to the callout request for proper processing of the response message.

You might also need to correlate the response message to the terminal or TCP/IP client application that originated the callout request.

Correlating a response to an asynchronous callout request

The IMS application programs are responsible for correlating the response message to the initial asynchronous callout request. An IMS application can define a data element in the callout request, such as a message identifier or a unique callout request ID, that can be used to correlate a response with the initial input message.

After a response has been correlated to its initiating callout request, the response can then be correlated to the original inputting terminal or IMS Connect TCP/IP client.

Routing asynchronous callout request responses to the inputting terminal

To route a response message to a terminal, the IMS application that issues the asynchronous callout request can capture the LTERM name of the terminal.

After it receives the response message, the IMS application that processes the response can then identify the originating terminal by the LTERM name. The LTERM name can either be included as part of the data of the callout request and the response messages or the LTERM name can be saved in a temporary database that is accessible to the IMS application that processes the response.

If a temporary database is used, the data used for the correlation of the response to the callout request must be included with the LTERM name in the database. After the response is returned, the IMS application can return the response to the initiating terminal by making a CHNG call and ISRT the response to the LTERM name captured by the IMS application program that issued initial callout request.

Routing asynchronous callout responses to an IMS Connect TCP/IP client

To route the response to the TCP/IP client, the IMS application program that issues the asynchronous callout request message must capture the IMS Connect TMEMBER and OTMA TPIPE names used by the initiating TCP/IP client.

Upon receipt of the response to the callout request, the IMS application program that processes the response uses the TMEMBER and TPIPE names to return the response to the correct TCP/IP client. To successfully route the response to the correct TMEMBER, you will also need to use either the OTMA destination descriptor or the OTMAYPRX user exit and DFSYDRU0 exit routine.

To make the TMEMBER and TPIPE names available to the IMS application program that processes the response, the names can be included as part of the data of the callout request and response messages or the names can be saved in a temporary database that is accessible to the IMS application program that processes the response.

If a temporary database is used, the IMS application program processing the response must use further correlation to retrieve the correct TMEMBER and TPIPE names when the response transaction is returned. When correlation is complete, the IMS application programs sends the response back to the original IMS Connect TCP/IP client by making a CHNG call and an ISRT using the TMEMBER and TPIPE names.

Callout and OTMA parallel processing of RESUME TPIPE requests

You can increase the throughput of outgoing callout request messages by enabling OTMA parallel processing of RESUME TPIPE requests.

By default, OTMA tpipes support only a single active RESUME TPIPE request at a time. If more RESUME TIPE requests are received, OTMA queues them to the tpipe and they remain in active until the active RESUME TPIPE request terminates. When the outgoing volume of callout messages is high, the OTMA tpipe can become a bottleneck.

When an OTMA tpipe supports multiple active RESUME TPIPE requests and multiple clients issue RESUME TPIPE requests to the tpipe, OTMA can send the callout messages to multiple clients in parallel.

Support for the parallel processing of RESUME TIPE requests can be enabled by specifying MULTIRTP=Y in an OTMA client descriptor. MULTIRTP=Y can be defined as the default for all OTMA clients by using the DFSOTMA system client descriptor.

A maximum number of active RESUME TPIPE requests can be set by specifying the LIMITRTP parameter in the OTMA client descriptors.

Support for parallel processing of RESUME TPIPE requests can also be enabled in IMS Connect by specifying MULTIRTP=Y in the definition of the data store connection to OTMA. MULTIRTP=Y can also be specified as the default for all connections from an IMS Connect in the system configuration of an IMS Connect instance.

Related concepts

“OTMA tpipe support for parallel processing of multiple active RESUME TPIPE requests” on page 769
When MULTIRTP=Y is specified in an OTMA client descriptor, the OTMA tpipes that are associated with the OTMA client can support multiple active resume tpipe requests in parallel, unless the MULTIRTP specification is overridden by the client.

“Retrieving output with parallel RESUME TPIPE requests” on page 327
To increase the throughput of OTMA output messages, especially callout request messages, you can enable an OTMA tpipe to support multiple active **RESUME TPIPE** requests in parallel by specifying MULTIRTP=Y in either the HWS or the DATASTORE IMS Connect configuration statement.

Chapter 46. OTMA message prefix

OTMA messages have a prefix that conforms to a format that is mapped by the DFSYMSG macro in the IMS.ADFSMAC data set.

The maximum length for a message prefix is 4096 bytes; this length does not include the application data.

The OTMA message prefix can contain the following sections:

1. Message control information section
2. State data section
3. Security data section
4. User data section
5. Application data section

The following table shows the sections of the OTMA message prefix and lists some of the key fields within each section. In this table, the term "Server" refers to IMS.

Table 154. OTMA message prefix segments and their key fields

Message control information	State data	Security data	User data	Application data
Tpipe name	Destination override	User ID		
Message type	Map name	Utoken		
Sequence numbers	Sync flags	Security flags		
Processing flag	Sync_level			
Response indicator	Commit mode tokens			
Chaining indicator	Server state			

Message-control information section

For every OTMA message, you must provide message-control information in the first section of the OTMA message prefix.

The message-control information section is mapped by the TMAMCTL DSECT of the DFSYMSG macro.

The following table is a summary of the content of the message-control information section of the message prefix (column 1 from the table in Chapter 46, “OTMA message prefix,” on page 859). The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 155. Format of the message-control information section of the OTMA message prefix

Byte	Length	Content	Value	Meaning
0	1	**Architecture level**	X'01'	OTMA release level. This is mandatory for all messages.
1	1	**Message type**: A specification in this field is mandatory for all messages.	X'80'	*Data*: Server output data (output from an IMS application program). This data is not a transaction.
			X'40'	*Transaction*: Transaction or IMS command input to the server. The actual transaction name is specified in the application-data section of the prefix.
			X'20'	*Response*: A response message.
			X'10'	*Protocol command*: An OTMA protocol command (not an IMS command). See "Protocol command type" below at byte offset 4.
			X'08'	*Commit Confirmation*: Commit complete. Used by the server to notify the client of sync point completion. Only used for send-then-commit transactions. See "commit-confirmation flag" below.
			X'04'	*Synchronous program switch*: A synchronous program switch request.

© Copyright IBM Corp. 1974, 2020

Table 155. Format of the message-control information section of the OTMA message prefix (continued)

Byte	Length	Content	Value	Meaning
2	1	**Response flag**: Values in the response flag field are mutually exclusive.	X'80'	*ACK*: Positive acknowledgment.
			X'40'	*NAK*: Negative acknowledgment. A NAK can be accompanied by an additional sense code.
			X'20'	*Response Requested*: A response is requested for this message.
			X'10'	*Extended Response Requested*: Requests architected transaction or command attributes to be returned to the client.
			X'08'	*Response to a synchronous callout request*: This message is a response to a synchronous callout request that was issued by an IMS application program.
			X'04'	*The input transaction expired before GU call*: Set by the server, this flag indicates that the input transaction timed out on the IMS input queue before an IMS application could issue a GU call to retrieve it.
			X'02'	*Support for delayed ACK or NAK response:* The OTMA client sets this flag to instruct OTMA to send a NAK to the client if OTMA receives a late or invalid ACK/NAK from the client.
			X'01'	*Return input message on transaction expiration*: Set by the client to request that, if a transaction times out before processing by IMS, OTMA returns the original input message to the client instead of message DFS3588I.
3	1	**Commit-confirmation and other flags**	X'80'	*Committed*: Server committed successfully.
			X'40'	*Aborted*: Server aborted commit.
			X'08'	*Aborted due to the OTMA timeout condition*: Indicates that an ACK or NAK response from the OTMA client was not received before the timeout limit was reached.
			X'04'	SENDALTP function activated for IMS Connect commit-then-send messages.
4	1	**Protocol command type**	X'04'	*Client-Bid*: Sent by a client to the server. The response-requested flag and the appropriate state data fields (for example, `Member Name`) must also be set.
			X'08'	*Server Available*: Sent by the server to a client. The appropriate state data fields (for example, `Member Name`) must also be set.
			X'0C'	*CBresynch*: Sent by a client to the server to request a resynchronization. This client-bid request with resynchronization to follow is optional, and causes the server to send an SRVresynch command to the client.
			X'14'	*Suspend Processing for All Tpipes*: The server sends this command to suspend all message activity with the client.
			X'18'	*Resume Processing for All Tpipes*: The server sends this command to resume message processing with the client.
			X'1C'	*Suspend Input for Tpipe*: The server sends this command when it is overloaded.
			X'20'	*Resume Input for Tpipe*: The server sends this command when it is ready for client input (following a Suspend Input for tpipe command).
			X'24'	*Resume output for tpipe*: Sent by a client to the server to request queued tpipe output be resent.
			X'26'	*Resume output for all tpipes*: Sent by client to request that OTMA resume sending output messages from all tpipes associated with the client. Does not apply to tpipe hold queues.
			X'28'	*Resume output for the hold queue for tpipe*: Sent by a client to the server to request messages from the hold queue for tpipe.
			X'29'	*Cancel resume tpipe request*: Sent by a client to the server to cancel a pending resume tpipe request. The resume tpipe request must be identified by specifying its resume tpipe token in byte 4 of the OTMA state data.
			X'2A'	*No messages on tpipe hold queue*: Sent by the server to the client when a resume tpipe request is received and the tpipe hold queue is empty.
			X'2C'	*SRVresynch*: Sent by the server to a client who has sent a CBresynch. This command identifies all synchronized tpipes within the server.
			X'30'	*REQresynch*: Sent by the server to a client to specify the state of a synchronized tpipe.
			X'34'	*REPresynch*: Sent by a client to the server to indicate the type of resynchronization to be performed by the server.
			X'38'	*TBresynch*: Sent by a client to the server to initiate resynchronization for a particular tpipe.
			X'3C'	*Server state*: Sent by the server to the client with the current status of server processing. The client can use this information to redirect transactions to a different server if server processing slows.

Table 156. Format of the message-control information section of the OTMA message prefix

Byte	Length	Content	Value	Meaning
5	1	**Processing flag**	X'80'	*Resume Tpipe token*: For resume tpipe requests only, indicates that a resume tpipe request includes a token that uniquely identifies the request. OTMA uses the token to queue and process multiple resume tpipe requests in order. The Resume Tpipe token is also specified by the client when the client cancels a resume tpipe request. For output messages, this flag indicates that the suspended processing for all tpipes (TMAMCSPA) is due to IMS shutdown. This is only an output flag.
			X'40'	*Synchronized Tpipe*: Input and output sequence numbers are maintained for the transaction pipe.
			X'20'	*Asynchronous output*: The server is sending unsolicited queued data messages.
			X'10'	*Error Message Follows*: An error data message follows. Set by the server when sending a NAK.
			X'08'	*Message in Hold Queue*: One or more messages in the hold queue for tpipe. Not available for shared queues.
			X'02'	Extra info set. For a RESUME TPIPE request, see byte 24 of the message-control information of the OTMA message prefix for the RESUME TPIPE ID info. For a callout message, see byte 22 in the transaction and callout messages of the state data section prefix of the OTMA message prefix for the RESUME TPIPE ID info.
			X'01'	*Error Message Sent*: An error message was sent as a CM1 response. A CM1 transaction generated an error message as the only response.

Table 157. Format of the message-control information section of the OTMA message prefix

Byte	Length	Content	Value	Meaning
6	8	**Tpipe name**		OTMA identification and processing control token. This name is used to override the LTERM name on the I/O PCB for an IMS application program.
14	1	**Chain flag** This flag is mandatory for multi-segment messages.	X'80'	*First-In-Chain*: The first segment of a multi-segment message. A message of only one segment is indicated by setting both the first-in-chain and last-in-chain flags.
			X'40'	*Middle-In-Chain*: Part of a multi-segment message.
			X'20'	*Last-In-Chain*: The last segment of a multi-segment message.
			X'10'	*Discard Chain*: Discard the current chain of message segments.
15	1	**Prefix flag** The value of this field indicates which sections of the message prefix are attached to this message.	X'80'	*State Data*: The state-data section is included with the message. State data section is mandatory for each message.
			X'40'	*Security*: The security section is included with the message.
			X'20'	*User Data*: The user-data section is included with the message. This data is specified by an OTMA client.
			X'10'	*Application Data*: The application-data section is included with the message.
16	4	**Send-sequence number**		The sequence number for the transaction pipe. Incremented on every send for each transaction pipe.
20	4	**Sense code**		
	ORG to byte offset 20 (TMAMCSNS)			
20	2	**Sense code**		Accompanies a NAK message.
22	2	**Reason code**		Accompanies a NAK message.
	ORG to byte offset 20 (TMAMCSNS)			
20	4	**Aging value**		Specifies the accessor environment element (ACEE) aging value, in seconds. IMS creates an ACEE if the age of the current ACEE is greater than this value. The aging value specifies how often the cached user ID ACEE should be refreshed. The aging value flag, X'40', must also be set in byte 5 of the State Data. This value does not apply to an ACK or NAK message. The minimum value for caching support is 300 seconds (5 minutes). If the aging value specified is less than the minimum, IMS always creates a non-cached ACEE.
24	4	**Recoverable sequence number**		The recoverable sequence number for the transaction pipe. Incremented on every send of a recoverable message using a synchronized transaction pipe. Required for resynchronization only.
28	2	**Segment sequence number**		Sequence number for segments of a multi-segment OTMA message.
30	1	**CM1 ACK Timeout**	1 to 255	Specifies a message level timeout value in seconds for this message.
31	1	**Reserved**		Applies only to send-then-commit messages when synclevel=confirm or synclevel=syncpt. Sets a message level timeout value for this transaction only. The value cannot be greater than the timeout value set in OTMA; otherwise, the value is ignored. The time out value is set in OTMA either by the **/START TMEMBER** command or the OTMA client descriptor.
		Number of control data segments	1 to 255	Specifies the number of control data segments when TMAMHCTD is set in the state data. This field is used only when IMS sends an ICAL callout message with control data.
	ORG to byte offset 24 (TMAMCRSQ)			

Table 157. Format of the message-control information section of the OTMA message prefix (continued)

Byte	Length	Content	Value	Meaning
24	8	**Resume tpipe requester ID**		

Related reference

“Explanation of OTMA message-control information fields” on page 862
The fields in the message-control information portion of the OTMA message prefix are used to specify the characteristics of an OTMA message, such as its type, contents, or processing requirements.

Explanation of OTMA message-control information fields

The fields in the message-control information portion of the OTMA message prefix are used to specify the characteristics of an OTMA message, such as its type, contents, or processing requirements.

This topic provides explanations for the fields in the message-control information section of the message prefix.

Architecture level (TMAMCALV)
Specifies the OTMA architecture level. The client specifies an architecture level, and the server indicates in the response message which architecture level it is using. The architecture levels used by a client and a server must match.

Mandatory for all messages.

Message type (TMAMCMGT)
Specifies the message type. Every OTMA message must specify a value for the message type. The values are not mutually exclusive. For example, when the server sends an ACK message to a client-submitted transaction, both the transaction and response flags are set.

Data (TMAMCDTA - X'80')
Specifies server output data sent to the client. If the client specifies synchronization level `Confirm` in the state-data section of the message prefix, the server also sets `Response Requested` for the response flag. If the client does not specify a synchronization level, the server uses the default, `Confirm`.

Transaction (TMAMCTXN - X'40')
Specifies client input data to the server.

Whether the server replies with an ACK or NAK message depends only on whether `Response Requested` is also set for the response flag.

Response (TMAMCRSP - X'20')
Specifies the message type as response message, and is set when the message response flag specifies `Response Requested`.

If this flag is set, the response flag specifies either ACK or NAK.

The send-sequence numbers must match for the original data message and the response message. Chained transaction input messages to the server must always request a response before the next transaction (for a particular transaction pipe) is sent.

Command (TMAMCCMD - X'10')
Specifies an OTMA protocol command. OTMA commands must always specify Response Requested for the Response flag.

Commit confirmation (TMAMCCMT - X'08')
Specifies that commit is complete. This is sent by the server when a sync point has completed, and is only applicable for send-then-commit transactions. This flag can also be set by an OTMA client to inform the OTMA server to end the IMS conversational transaction.

Response flag (TMAMCRSI)
Specifies either that the message is a response message or that a response is requested.

Acknowledgments to transactions include attributes (for that transaction) in the application-data section of the message prefix only if the transaction specifies Extended Response Requested.

ACK (TMAMCACK - X'80')
Specifies a positive acknowledgment.

NAK (TMAMCNAK - X'40')
Specifies a negative acknowledgment.

See the sense code field for more information on the reason for the NAK.

Response requested (TMAMCRRQ - X'20')
Specifies that a response is requested for this message. This can be set for message types of Data, Transaction, or Command.

When sending send-then-commit IMS command output, IMS does not request an ACK regardless of the synchronization level.

Extended response requested (TMAMCERQ - X'10')
Specifies that an extended response is requested for this message. Can be set by a client only for transactions (or for transactions that specify an IMS command instead of a transaction code).

If this flag is set for a transaction, IMS returns the architected attributes for that transaction in the application-data section of the ACK message.

If this flag is set for a command, IMS returns the architected attributes in the application-data section of the ACK message. This flag can be set for the IMS commands **/DISPLAY TRANSACTION** and **/DISPLAY TRANSACTION ALL**.

Response to a synchronous callout request (TMAMSYRP - X'08')
Specifies that this message is a response to a synchronous callout message issued by an IMS application program running in an IMS dependent region.

When the flag for a synchronous callout response is set, most hold-queue capable clients, such as IMS Connect, also set the send-only message flag (TMAMHSOM - X'80') in the client flag field (TMAMHCFL) of the state data section of the message prefix.

If both the response-requested flag (TMAMCRRQ - X'20') and the TMAMSYRP flag are set in the response flag field, OTMA sends the client an indication (an ACK or NAK) of whether the response was successfully delivered to the IMS ICAL application.

Support for delayed ACK or NAK response (TMAMDACK - X'02')
Specifies that the client asks OTMA to return a NAK response to the client if OTMA receives a late or invalid acknowledgement from the client. For example, OTMA returns a NAK to the client when OTMA receives an acknowledgement after either the ACK timeout interval expires or a **/STO TMEMBER** TPIPE command has been issued clear the WAIT status of the tpipe. The NAK returned to the client by OTMA includes an X'2B' sense code.

Commit-confirmation and other flags
Specifies the success of a commit request. Sent by the server to the client in a commit-confirmation message. These messages are only applicable for send-then-commit transactions, and are not affected by the synchronization-level flag in the state-data section of the message prefix.

Committed (TMAMCCTD - X'80')
Specifies that the server committed successfully.

Aborted (TMAMCABT -X'40')
Specifies that the server aborted the commit.

Committed (TMAMCRTC - X'20')

Specifies that the server is ready to commit the output in the IOPCB after the server receives a commit notification from the client via RRS.

Aborted due to the OTMA timeout condition (TMAMCTMO -X'08')
Indicates that an ACK or NAK response from the OTMA client was not received before the timeout limit was reached.

Message level activation for SENDALTP function (TMAMALTP -X'04')
Indicates that a message level activation of the SENDALTP function for IMS Connect is specified.

Command type (TMAMCTYP)
Specifies the OTMA protocol command type.

IMS MTO commands are specified in the application-data section of the message.

Client-bid (TMAMCBID - X'04')
Specifies the first message a client sends to the OTMA server. This command must also set the response-requested flag and the security flag in the message-control information section of the message prefix. The appropriate state-data fields (for example, Member Name) must also be set.

The security-data prefix must specify a Utoken field so the OTMA server can validate the client's authority to act as an OTMA client.

Because the server can respond to the client-bid request, this message should not be sent until the client is ready to start accepting data messages.

Server available (TMAMCAVL - X'08')
Specifies the first message the server sends to a client. It is sent when the server has connected to the z/OS cross-system coupling facility (XCF) group before the client has connected. The client replies to the Server Available message with a client-bid request. The appropriate state data fields (for example, `Member Name`) must also be set.

If the client connects first, it is notified by XCF when the server connects, and begins processing with a client-bid request.

CBresynch (TMAMCRSN - X'0C')
Specifies a client-bid message with a request by the client for resynchronization. This command is optional and causes the server to send an SRVresynch message to the client. The CBresynch command is the first message that a client sends to the OTMA server when it attempts to resynchronize with IMS and existing synchronized tpipes exist for the client. Other than the CBresynch message indicator in the message prefix, the information required for the message prefix should be identical to the client-bid command.

If IMS receives a client-bid request from the client and IMS is aware of existing synchronized tpipes, IMS issues informational message DFS2394I to the MTO. IMS resets the recoverable send- or receive-sequence numbers to 0 (zero) for all the synchronized tpipes.

Suspend processing for all tpipes (TMAMCSPA - X'14')
Specifies that the server is suspending all message activity with the client. All subsequent data input receives a NAK message from the server. Similarly, the client should send a NAK message for any subsequent server messages.

Clients can suspend processing for a particular transaction pipe by submitting a **/STOP TPIPE** command as an OTMA message.

Resume processing for all tpipes (TMAMCRSA - X'18')
Specifies that the server is resuming message activity with the client.

Clients can resume processing for a particular transaction pipe that has been stopped by submitting a **/START TPIPE** command as an OTMA message.

Suspend input for tpipe (TMAMCSPN - X'1C')
Specifies that the server is overloaded and is temporarily suspending input for the transaction pipe. All subsequent client input receives NAK messages for the transaction pipe specified in the message-control information section of the message prefix. A response is not requested for this command.

This command is also sent by IMS when the master terminal operator enters a **/STOP TPIPE** command.

Resume input for tpipe (TMAMCRSM - X'20')
Specifies that the server is ready to resume client input following an earlier Suspend Input for tpipe command. A response is not requested for this command.

This command is also sent by IMS when the IMS master terminal operator issues a **/START TPIPE** command.

Resume output for tpipe (TMAMCRTP - X'24')
Specifies one or multiple tpipe names to the OTMA server. All queued output on the tpipes will be resent again.

Resume output for all tpipes (TMAMCRAT - X'26')
Sent by client to request that OTMA resume sending output messages from all tpipes associated with the client. Does not apply to tpipe hold queues.

Resume output for the hold queue for tpipe (TMAMCRHQ - X'28')
Specifies that a client is requesting to retrieve messages from the hold queue for tpipe. There are command options to retrieve messages.

Cancel resume tpipe request (TMAMCDRH - X'29')
Sent by a client to the server to cancel a pending resume tpipe request. The resume tpipe request must be identified by specifying its resume tpipe token in byte 4 of the OTMA state data.

No messages on tpipe hold queue (TMAMCMSG - X'2A')
Sent by the server to the client when a resume tpipe request is received and the tpipe hold queue is empty.

SRVresynch (TMAMCSRS - X'2C')
Specifies the server's response to a client's CBresynch command. This command specifies the states of synchronized transaction pipes within the server (the send- and receive-sequence numbers).

This command is sent as a single message (with single or multiple segments), and an ACK is requested.

REQresynch (TMAMCRQS - X'30')
Specifies the send-sequence number and the receive sequence for a particular tpipe. REQresynch is sent from IMS to a client.

REPresynch (TMAMCRPS - X'34')
Specifies the client's desired state information for a tpipe. A client sends the REPresynch command to IMS in response to the REQresynch command received from IMS.

TBresynch (TMAMCTBR - X'38')
Specifies that the client is ready to receive the REQresynch command from IMS.

Resource state (TMAMMNTR - X'3C')
Sent by server to notify the client of the state of OTMA resources. The client can use this information to redirect transactions to a different server if server processing slows.

Processing flag (TMAMCPFG)
Specifies options by which a client or server can control message processing.

Resume tpipe token (TMAMQRTP - X'80')
For resume tpipe requests only, indicates that a resume tpipe request includes a token that uniquely identifies the request. OTMA uses the token to queue and process multiple resume tpipe requests in order.

The Resume Tpipe token is also specified by the client when the client cancels a resume tpipe request.

Synchronized tpipe (TMAMCSYP - X'40')
Specifies that the transaction pipe is to be synchronized. Allows the client to resynchronize a transaction pipe if there is a failure. Only valid for commit-then-send transactions.

This flag causes input and output sequence numbers to be maintained for the transaction pipe. All transactions routed through the transaction pipe must specify this flag consistently (either on or off).

Asynchronous output (TMAMCASY - X'20')
Specifies that the server is sending unsolicited queued output to the client. This can occur when IMS inserts a message to an alternate PCB.

Certain IMS commands, when submitted as commit-then-send, can cause IMS to send the output to a client with this flag set. In this case, the OTMA prefixes contain no identifying information that the client can use to correlate the output to the originating command message. These command output data messages simply identify the transaction-pipe name. IMS can also send some unsolicited error messages with only the transaction-pipe name.

Error message follows (TMAMCERR - X'10')
Specifies that an error message follows this message. This flag is set for NAK messages from the server. An additional error message is then sent to the client.

The asynchronous-output flag is not set in the error data message, because the output is not generated by an IMS application.

Message in the hold queue (TMAMCQUE - X'08')
Specifies that one or more messages exist in the hold queue for the tpipe to be delivered. This flag is always on for an IMS output message that has been sent from the hold queue for tpipe. Therefore, this flag can be used to determine whether there is any message in the hold queue for an IMS output message that has been sent from the regular queue for tpipe.

To determine whether the IMS output message is sent from the regular queue or from the hold queue, check the "From Hold Queue" flag in the Server State of the State Data.

To retrieve one or more messages from the hold queue, issue the "Resume Output for the Hold Queue for tpipe" protocol command.

Extra info set (TMAMXINF - X'02')
For a RESUME TPIPE request, it indicates that the user ID of the client who issues the request is included in the last 8-byte of the message-control information section. See Resume tpipe requester ID (TMAMRTID) below.

For a callout message, it indicates that the user ID of the client who issues the request is included in the state data section at offset 22. See Resume tpipe user ID (TMAMRTOD) in Transaction and callout messages (Communications and Connections).

Error message sent (TMAMCER3 - X'01')
An error was detected while processing a CM1 transaction and the application did not issue an ISRT to the IOPCB. Therefore, the DFS message is the only response for this CM1 transaction. The OTMA client, for example IMS Connect or IBM MQ, must deliver the DFS message to the client application even if OTMA sets the deallocate abort flag in the commit confirmation message.

IMS Shutdown in progress (OMCTSHDN - X'80')
This flag indicates that the suspended processing for all tpipes (TMAMCSPA) is due to IMS shutdown.

This is only an output flag.

Tpipe name (TMAMCTNM)
An 8-byte character field that specifies a transaction-pipe name. For IMS, this name is used to override the LTERM name on the I/O PCB. This field is applicable for all transaction, data, and commit-confirmation message types. It is also applicable for certain response and command message types.

Chain flag (TMAMCCHN)
Specifies how many segments are in the message. This flag is applicable to transaction and data message types, and it is mandatory for multi-segment messages.

First-in-chain (TMAMCFIC - X'80')
Specifies the first segment in a chain of segments which comprise a multi-segment message. Subsequent segments of the message only need the message-control information section of the message prefix. Other applicable prefix segments (for example, those specified by the client on the transaction message) are sent only with the first segment (with the first-in-chain flag set).

If the OTMA message has only one segment, the last-in-chain flag should also be set.

Middle-in-chain (TMAMCMIC - X'40')
Specifies a segment that is neither first nor last in a chain of segments that comprise a multi-segment message. These segments only need the message-control information section of the message prefix.

Restriction: Because the client and server tokens are in the state-data section of the message prefix, they cannot be used to correlate and combine segmented messages. The transaction-pipe name and send-sequence numbers can be used for this purpose; they are in the message-control information section of the message prefix for each segment.

Last-in-chain (TMAMCLIC - X'20')
Specifies the last segment of a multi-segment message.

Discard chain (TMAMCCAN - X'10')
Specifies that the entire chain of a multi-segment message is to be discarded. The last-in-chain flag must also be set.

Prefix flag (TMAMCPFL)
Specifies the sections of the message prefix that are attached to the OTMA message. Every message must have the message-control information and state-data sections, but any combination of other sections can be sent with an OTMA message.

State data (TMAMCSTD - X'80')
Specifies that the message includes the state-data section of the message prefix.

Security data (TMAMCSEC - X'40')
Specifies that the message includes the security-data section of the message prefix.

User data (TMAMCUSR - X'20')
Specifies that the message includes the user-data section of the message prefix.

Application data (TMAMCAPP - X'10')
Specifies that the message includes the application-data section of the message prefix.

Send-sequence number (TMAMCSSN)
Specifies the sequence number for a transaction pipe. This sequence number is updated by the client and server when sending messages or transactions.

Recommendation: Increment the number separately for each transaction pipe.

This number can also be used to match an ACK or NAK message with the specific message being acknowledged.

Sense code (TMAMCSNS)
Specifies a 4-byte sense code that accompanies a NAK message. TMAMCSNS has two parts, a 2-byte sense code (TMAMCSNC) and a 2-byte reason code (TMAMCRSC). For an explanation of the codes returned in this field, see *IMS Version 15 Messages and Codes, Volume 2: Non-DFS Messages*.

Sense code (TMAMCSNC)
Specifies a 2-byte sense code that accompanies a NAK message. For an explanation of the codes returned in this field, see *IMS Version 15 Messages and Codes, Volume 2: Non-DFS Messages*.

Reason code (TMAMCRSC)
Specifies the 2-byte reason code that accompanies a NAK message. This code can further qualify a sense code.

Userid aging value (TMAMAGNG)
Specifies a 4-byte aging value in seconds for the input user ID. This field is different from the aging value specified in the OTMA client-bid command for OTMA connection. The aging value in the client-bid command sets the default aging value for all the OTMA user IDs; however, the Userid Aging Value overrides the default aging value for a specific user ID. If the Userid Aging Value is less than 300 seconds (5 minutes), IMS always creates a non-cached accessor environment element (ACEE).

Recoverable sequence number (TMAMCRSQ)
Specifies the recoverable sequence number for a transaction pipe. Incremented on every send of a recoverable message using a synchronized transaction pipe. Both the client and the server increment

their recoverable send-sequence numbers and maintain them separately from the send-sequence number. Required for resynchronization only.

Segment sequence number (TMAMCSEQ)
Specifies the sequence number for a segment of a multi-segment message. This number must be updated for each segment, because messages are not necessarily delivered sequentially by XCF.

This number must have a value of 1 if the message has only one segment.

Resume tpipe requester ID (TMAMRTID)
The user ID of the client that is issuing the resume tpipe request.

Related concepts

"Client/server resynchronization with OTMA" on page 837
In order to guarantee that client transactions are processed and that they are processed only once, OTMA provides a protocol for synchronizing transactions.

Related reference

"OTMA resynchronization protocol" on page 840
OTMA resynchronization is based on a series of command exchanges with each client.

"State data section" on page 868
The state data is mandatory for any OTMA message. It immediately follows the message-control information section in the message prefix. It contains transaction-related information.

"Security data section" on page 883
The security-data section is mandatory for every transaction or command, and is optional for OTMA protocol commands.

"User data section" on page 886
The user-data section of the OTMA message prefix is variable length and follows the security-data section. It can contain any data.

"Application data section" on page 886
The application-data section of OTMA messages is variable length and follows the user-data section of the message prefix.

"Transaction and callout messages" on page 874
The state data for the transaction-related and synchronous callout request information in the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

"Resume output for the hold queue for tpipe" on page 881
The state data for resume output for the hold queue for tpipe portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

State data section

The state data is mandatory for any OTMA message. It immediately follows the message-control information section in the message prefix. It contains transaction-related information.

The state-data section has different formats for transaction-related information and for OTMA protocol commands. The state data section can be followed by a security-data section.

Server-Available and Client-Bid commands

The state data for the Server-Available and Client-Bid commands section of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The following table summarizes the format for state data for command messages, including byte offset, length, content, hexadecimal value, the meaning, and usage comments.

Table 158. Server-Available and Client-Bid command format

Byte	Length	Content	Value	Meaning
0	2	**Length**		Specifies the total length of the state-data section of the message prefix, including the length field.
2	16	**Member name**		Specifies the z/OS cross-system coupling facility (XCF) member name of originating server.
18	8	**Originator's token**		Specifies the XCF-member token of the originator (either client or server) of the message.
26	8	**Destination token**		Specifies the XCF-member token of the destination (either client or server) of the message.
Note: The following fields are present only for client-bid commands.				
34	8	**DRU exit name**		Specifies the name of the OTMA Destination Resolution exit routine.
42	2	**MaxBlocksize**		Specifies the maximum block size for XCF conversations between the server and the client. This field is optional
44	1	**Client-Bid flag**	X'80'	*Hold Queue:* Specifies that a hold queue for a tpipe is needed. The hold queue can hold commit-then-send output that is receives a NAK response, as well as alternate PCB output for an OTMA client. The output messages in the queue will not be delivered until the client requests that those messages be delivered. Use of the hold queue is optional. Without the hold queue, a regular queue for the tpipe will be used to hold and deliver all the Commit-then-send output messages. The default is no hold queue for a tpipe. OTMA resynchronization protocol currently does not support the hold queue.
			X'40'	Reserved for OTMA Callable Interface.
			X'20'	Reserved for IMS Connect.
			X'10'	Reserved for IBM MQ.
			X'08'	Reserved for synchronous program switch.

Table 158. Server-Available and Client-Bid command format (continued)

Byte	Length	Content	Value	Meaning
45	1	**Additional client-bid flags**	X'80'	*Message flood control:* specifies that OTMA use the maximum number of input messages provided by the client-bid command at byte 62 for message flood control.
			X'20'	*Member timeout:* specifies that OTMA use the member timeout value for send-then-commit messages provided by the client-bid command at byte 65.
			X'10'	*Cascaded transaction support:* specifies support for cascading global RRS transactions (synchlevel=2 or syncpoint) from an OTMA client on one LPAR to OTMA on another LPAR.
			X'08'	*Super member flag:* activates the super member function for this member or client. To use this optional super member function, the member or client needs to set this flag and to specify the super member name, see the super member name at byte 54 below.
			X'04'	*Synchronous callout support:* specifies support for synchronous callout messages that are issued by IMS applications.
			X'02'	*CM0 timeout queue name:* Set CM0 timeout queue name.
			X'01'	*IMS-to-IMS TCP/IP communications:* Set by IMS Connect to indicate that IMS Connect supports IMS-to-IMS TCP/IP communications.
46	4	**Aging value**		Specifies the accessor environment element (ACEE) aging value in seconds. If the age of an existing ACEE for a user ID is greater than this value, IMS creates a new ACEE. If the aging value specified is less than the supported minimum value, IMS creates a non-cached ACEE. The minimum value for caching support is 300 seconds (5 minutes).
50	4	**Hash table size**		Defines the size of the IMS OTMA hash table for processing multi-segment messages input. The hash table is used to correctly chain all the input segments together. The suggested value is X'00000065'.
54	4	**Super member name**		Specifies a 1-4 byte super member name. When the super member function is activated, all of the asynchronous output messages for hold queue capable member or client, such as IMS Connect, will be shared by a set of clients specifying the same super member name.

Table 158. Server-Available and Client-Bid command format (continued)

Byte	Length	Content	Value	Meaning
58	2	**Offset to callout token**		Specifies the offset in the user data section at which this client expects the synchronous callout token to be placed by OTMA.
60	2	**Offset to remote destination definitions**		Specifies the offset in the user data section at which the definitions that are required for delivering the message to a remote IMS system can be found.
62	2	**Message flood threshold value**	0-9999	Specifies the maximum number of concurrent input messages from this member or client. If 0 is specified, the OTMA default is 5000. Values between 1 and 200 are treated as 200.
64	1	**Client-bid flags**	X'80'	MULTIRTP=Y is set for this tpipe. The tpipe supports multiple active resume tpipe requests.
			X'40'	MULTIRTP=N is set for this tpipe. The tpipe supports only one active resume tpipe request at a time.
			X'10'	SENDALTP=Y is set for the IMS Connect data store. The SENDALTP function is activated. IMS delivers an ALTPCB output without RESUME TPIPE calls for IMS Connect.
			X'08'	SENDALTP=N is set for the IMS Connect data store. The SENDALTP function is not activated. IMS queues an ALTPCB output to an OTMA TPIPE hold queue until a RESUME TPIPE call is issued to retrieve it.
65	1	**Timeout value for acknowledgments from client**		Specifies the member level timeout value in seconds for both commit-then-send (CM0) and send-then-commit (CM1) OTMA messages. For CM1 messages, this timeout value applies only when synclevel=confirm or synclevel=syncpt. Valid timeout intervals are 1 to 255 seconds. If no time out value is specified, the OTMA default is 120 seconds. The timeout value specified in this field cannot be greater than the timeout value set OTMA; otherwise, the value is ignored. The time out value is set in OTMA either by the /START TMEMBER command or the OTMA client descriptor.
66	8	**CM0 timeout queue name**		Specifies an 8-byte name for a tpipe timeout queue for commit-then-send (CM0) output.

Related reference

"OTMA state data fields used by IMS Connect" on page 257

The tables in this topic describe the fields of the OTMA state data header and the order of those fields.

SRVresynch command

The state data for the SRVresynch command portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The SRVresynch command is sent by IMS to pass all its known synchronized tpipe names to the client. If the command data can not fit into a single buffer, chained multi-segment buffers will be sent instead. The following table summarizes the format of state data for the SRVresynch command. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 159. SRVresynch command format

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section, including the length field.
2	8	**Tpipe name**	The transaction pipe name. The tpipe name can be repeated as necessary.

REQresynch command

The state data for the REQresynch command portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The REQresynch command is used by IMS to pass the send-sequence number and the receive sequence for a specific tpipe to the client. The following table summarizes the format of state data for the REQresynch command. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 160. REQresynch command format

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section, including the length field.
2	8	Tpipe Name	The transaction pipe name.
10	4	**Send-sequence number**	IMS recoverable send-sequence number for the transaction pipe.
14	4	**Receive-sequence number**	IMS recoverable receive-sequence number for the transaction pipe.
18	1	**Tpipe flag 1**	Reserved for future use.
19	1	**Tpipe flag 2**	Reserved for future use.
20	6	**RESERVED**	

REPresynch command

The state data for the REPresynch command portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The REPresynch command is sent by the client in reply to the REQresynch request from IMS. It contains the desired state information for a tpipe. The following table summarizes the format of state data for the REPresynch command. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 161. REPresynch command format

Byte	Length	Content	Value	Description
0	2	**Length**		Length of the state-data section, including the length field.
2	8	**Tpipe name**		The transaction pipe name.
10	4	**Send-sequence number**		Client recoverable send-sequence number for the transaction pipe.
14	4	**Receive-sequence number**		Client recoverable receive-sequence number for the transaction pipe.
18	1	**Tpipe flag 1**	X'00'	*Continue:* the last message has been received and IMS continues processing for this synchronized transaction pipe.
			X'04'	*Dequeue Last Output:* IMS can dequeue the last output message. The recoverable send-sequence number is updated.
			X'08'	*Reset Sequence Numbers:* IMS resets the recoverable send-sequence number and the recoverable receive-sequence number, as passed in this command.
			X'0C'	*Stop Tpipe:* IMS stops this synchronized transaction pipe.
			X'10'	*Stop Tpipe and wait for TBresynch:* IMS stops this synchronized transaction pipe and waits for TBresynch from the client.
19	1	**Tpipe flag 2**		Reserved.
20	6	**Reserved**		

TBresynch command

The state data for the TBresynch command portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The TBresynch command is sent by the client to IMS if the client decides it is ready to receive REQresynch from IMS. The TBresynch command can be issued in the following two situations:

- The client has received an ACK message after sending REPresynch with "stop and wait for TBresynch" to IMS.
- The client may request a TBresynch with IMS at any time after the initial nondeferred resynchronization has completed for this tpipe.

The following table summarizes the format of state data for the TBresynch command. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 162. TBresynch command format

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section, including the length field.
2	8	**Tpipe name**	The transaction pipe name.

Transaction and callout messages

The state data for the transaction-related and synchronous callout request information in the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The following table summarizes the format of state data for transaction-related and synchronous callout request information. The summary includes byte, length, content, hexadecimal value, and descriptions of the use of each field.

Table 163. State data format for transaction-related information

Byte	Length	Content	Value	Description
0	2	**Length**		Specifies the total length of the state-data section of the message prefix, including the length field.
2	1	**Server state:** specifies the mode in which the transaction is running.	X'80'	*Conversational State:* for a conversational transaction. Both the server and the client set this flag when sending conversational data.
			X'40'	Set by the client for a commit-then-send (CM0) transaction. When this flag is set for a commit-then-send transaction, if the IMS application does not reply to the IOPCB or complete a message switch to another transaction, OTMA issues a DFS2082 message to the client, regardless of the transaction response mode. This flag will be cleared when the SENDALTP function is activated.
			X'20'	*From Hold Queue*: specifies that the output message was sent from the IMS hold queue for the tpipe. The server initially sets this flag when sending a commit-then-send output message. The client also needs to set this flag when sending the subsequent ACK or NAK to IMS.
			X'10'	*Hold queue capable:* asynchronous output can be sent to hold queue. Set by the server.
			X'08'	*Rerouted message:* output has been rerouted. Set by the server when processing a rerouted output.
			X'04'	*CM1 time out:* a time out occurs for this CM1 output. Set by the server.
			X'02'	This value has different meanings for different message types. *Transaction expiration: length of time:* On transaction messages, when X'02' is specified, the transaction expiration function is enabled and submitted transactions expire after a specified length of time. The length of time is specified in the TMAMOSXP field. *Network security information is included:* On callout messages to clients that issued a RESUME TPIPE call, X'02' specifies that the message contains network security information in the security-data section.
			X'01'	This value has different meanings for different message types. *Transaction expiration: point in time:* On transaction messages from the client, X'01' enables the transaction expiration function and indicates that the submitted transaction expires at a specified point in time. Specify the point in time in STCK format in the user data prefix. Specify the byte offset of the time specification at byte 22 (the TMAMOSXP field) of the state data for transaction messages. *Resume tpipe token included:* For messages sent by the server from a tpipe hold queue, X'01' signifies that the output message includes a resume tpipe token at byte 48 (TMAMRTOK) in the OTMA state data prefix.

Table 163. State data format for transaction-related information (continued)

Byte	Length	Content	Value	Description
3	1	**Synchronization or callout flag:** specifies either the commit mode of a transaction or that a message is a synchronous callout request.	X'80'	*TMAMHCTD:* This callout message has control data in the application data section of the message.
			X'40'	*Commit-then-send:* a commit-then-send (CM0) transaction. The server commits output before sending it to the client.
			X'20'	*Send-then-commit:* a send-then-commit (CM1) transaction. The server sends output to the client before committing it.
			X'10'	*SendOnly ACK no msg resp:* for response messages to synchronous callout requests, which are sent to OTMA with the Send Only with ACK protocol, this value indicates that the ACK to the response message does not include the response data also.
			X'08'	*Synchronous callout message:* a synchronous callout message originating from an IMS application that is running in an IMS dependent region.
			X'04'	*IMS-to-IMS TCP/IP communications message:* Set by IMS Connect to indicate that IMS Connect supports IMS-to-IMS TCP/IP communications.
			X'02'	*Need dump for transaction expiration*: Set by client to request a symptom dump and a DFS554A message if this transaction expires on the IMS input queue before an application issues a GU call to retrieve it.
			X'01'	*Send SQ BE ALTPCB msg to FE IMS*:: Set by the client to request that the output generated on a back-end IMS system in a shared queues environment be returned to the client via the front-end IMS system.
4	1	**Synchronization level:** specifies the transaction synchronization level; that is, the way in which the client and an IMS application program interact for program output messages. The default is Confirm, in which IMS always requests a response when sending commit-then-send output to a client.	X'00'	*None:* specifies that the programs participate in coordinated commit processing on resources updated during the conversation under the z/OS Resource Recovery Services recovery platform. The server application program does not request an ACK message when it sends output to a client. None is only valid for send-then-commit transactions.
			X'01'	*Confirm:* specifies that synchronization is requested. The server sends transaction output with the response flag set to `Response Requested` in the message-control information section of the message prefix. Confirm can be used for either commit-then-send or send-then-commit transactions.
			X'02'	*Syncpt:* specifies that the programs participate in coordinated commit processing on resources updated during the conversation under the RRS recovery platform. A conversation with this level is also called a protected conversation and the resources updated under this conversation use the two-phase commit protocol.

Table 163. State data format for transaction-related information (continued)

Byte	Length	Content	Value	Description
5	1	**Client flags:** specifies optional processing requested by the client.	X'80'	*Send Only Message:* this is a send only message; the response is placed on the hold queue. This flag is valid only if the client requested a hold queue during open.
			X'40'	*Aging Value or NAK message for callout* If specified for the aging value, this flag instructs IMS to accept the accessor environment element (ACEE) aging value specified at byte 20 of the message control information. If specified on a NAK response to a synchronous callout message, OTMA keeps the callout message on the queue until it is retrieved by another resume tpipe call or the callout message times out.
			X'20'	*Reroute Request:* reroute requested. Setting this flag reroutes CM0 output to the destination that is specified in the **Destination Override** field.
			X'10'	*Purge CM0 IOPCB output:* when CM0 IOPCB output cannot be delivered to the client, delete it. Set by the client in the input message.
			X'08'	*Enable message level time out:* enable the timeout value specified in the message-control information section. This timeout value only applies to this particular CM1 message. If there is no message level time out specified, the member level timeout value is used.
			X'04'	Obsolete. EWLM is no longer supported by IMS. If this flag is specified, it is ignored by OTMA.
			X'02'	*Ignore PURG Calls:* for commit then send (CM0) messages that generate multiple PURG calls in the TP PCB, when this optional flag is set, IMS ignores the PURG calls so that the OTMA client application receives one response CM0 message with multiple output segments. When this flag is not set, if there are multiple ISRT and PURG calls on the TP PCB for a CM0 message, the OTMA client application receives multiple response messages. This flag is applies only to CM0 input messages. If this flag is specified in a CM1 input message, IMS resets the flag and ignores it for any remaining program-to-program switches and ALT-PCB processing.
			X'01'	*Send only ordered:* identifies the protocol as send only with serial delivery. When the SENDALTP function is activated and an ALTPCB output is sent to an IMS Connect client, this flag is set on the output message for serviceability.
6	8	**Map name**		Specifies the formatting map used by the server to map input or output data streams (for example, 3270 data streams). Although OTMA does not provide MFS support, you can use the map name to define the output data stream. The name is an 8-byte MOD name that is placed in the I/O PCB. IMS replaces this field in the prefix with the map name in the I/O PCB when the message is inserted. The map name is optional.
14	16	**Server token**		For CM1 messages, the server sets the server token for correlation purposes. After receiving a CM1 output message from the server, the client must return the server token on the following acknowledgment (ACK or NAK) or conversational iteration.
Alternate mapping for bytes 14 through 29 when server token is not specified or not relevant				

Table 163. State data format for transaction-related information (continued)

Byte	Length	Content	Value	Description
14	4	**Super member name**		On output from ALTPCB, specifies the super member name for the OTMAYPRX user exit and DFSYDRU0 exit routine.
18	2	**Transaction expiration value or offset**		If the server state flag is set to X'02' (TMAMTXP2), this field contains the length of time in seconds after which the input transaction expires. If the server state flag is set to X'01' (TMAMTXP1), this field contains the offset to the point in time specification, in STCK format, in the user data prefix. This field cannot be used for CM1 conversational iterations.
20	2	**Offset to correlation token for synchronous callout messages**		Specifies the offset in the user data section at which the client can find the correlation token in a synchronous callout message. This value should match the offset specified by the client in the TMAMOSYN field of the state data in the client bid message.
22	8	**Resume tpipe user ID**		Specifies the user ID of the client who issues the RESUME TPIPE request.
End of alternate mapping for bytes 14 through 29				
30	16	**Correlation token**		A client token to correlate input with output. This token is optional and is not used by the server. **Recommendation:** Use this token to help clients manage their transactions.
46	16	**CM1 context ID**		On input transaction messages sent by the client, specifies the RRS token that is used with SYNCLVL=02 and protected conversations.
Alternate mapping for bytes 46 through 61 when CM1 context ID is not used				
46	8	**Resume tpipe token**		On output messages sent by the server from the tpipe hold queue, this field contains the resume tpipe request token.
54	8	**Time received from XCF**		In the message returned to a client after a transaction has expired, this field contains the time that the expired transaction was received by IMS from XCF.
End of alternate mapping for bytes 46 through 61				
54	8	**Program name for callout**		In an IMS synchronous callout message which is sent from an IMS dependent region, this field identifies the program that makes the DL/I ICAL call.
62	8	**Destination override**		Specifies an LTERM name used to override the LTERM name in the IMS application program's I/O PCB. This override is used if the client does not want to override the LTERM name in the I/O PCB with the transaction-pipe name. This optional override is not used if it begins with a blank.
70	2	**Server user data length**		Specifies the length of the server user data, if any. The maximum length of the server user data is 256 bytes. The server user data length is not included in the length calculation.
72	*	**Server user data**		Specifies any data needed by the server. If included in a transaction message by the client, it is returned by the server in the output data messages. Variable length. Optional.

Server state protocol command

The state data for the server state protocol command is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The server state protocol command is used by OTMA both to notify the OTMA client about the current state or a change in the state of IMS processing and as a heartbeat message, which is issued at 60 second intervals.

The server state protocol command is identified by X'3C' in the command type field (TMAMCTYP) of the message control information section of the OTMA header.

The following table summarizes the format of state data for the server state protocol command. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 164. Server state protocol command format

Dec offset	Hex offset	Length	Field name	Value and description
0	X'00'	2	TMAMHLEN	Length of the state-data section, including the length field itself.
2	X'02'	2	TMAMRSIM_STATUS	The state of OTMA resources in the server. **X'03' - Normal state** IMS is available and processing OTMA messages normally. **X'02' - Degraded state** IMS is processing OTMA messages slowly. OTMA issues a degraded state protocol command when one or more conditions indicate that IMS is not processing OTMA messages as quickly as it should. **X'01' - Unavailable state** IMS can no longer accept OTMA transactions for processing. OTMA issues the unavailable state protocol command to alert the OTMA client that one or more severe conditions prevent IMS from processing OTMA messages.
4	X'04'	1	TMAMRSIM_SVRFLG1	Flags for resource in the first group of unavailable resources. **X'80' - TMAMRSIM_S1SHTDN** The IMS server is shutting down and no longer available.
5	X'05'	1	TMAMRSIM_SVRFLG2	Reserved
6	X'06'	1	TMAMRSIM_SVRFLG3	Reserved
7	X'07'	1	TMAMRSIM_SVRFLG4	Flags for resource in the fourth group of unavailable resources. **X'01' - TMAMRSIM_S4FLOOD** The server is flooded with OTMA messages that are waiting to be processed and is no longer available.

Table 164. Server state protocol command format (continued)

Dec offset	Hex offset	Length	Field name	Value and description
8	X'08'	1	TMAMRSIM_WRNFLG1	Flags for resources in the first group of degraded resources. **X'80' - TMAMRSIM_W1FLOOD** Global flood warning for all OTMA clients **X'40' - TMAMRSIM_W1MTP** Warning for OTMA clients that specify a maximum allowable number of tpipes: the total number of tpipes has reached the global warning threshold defined by the highest value specified on the MAXTP parameter of any OTMA client descriptor. **X'20' - TMAMRSIM_W1MTPF** The MAXTP limit defined by the DFSOTMA client descriptor entry was reached. Subsequent requests for new tpipes from all OTMA clients are rejected. An exception to this is when MAXTPBE=NO is defined for a backend IMS system for application GU processing in a shared queues environment. **X'10' - TMAMRSIM_W1MTP80** The number of tpipes in the IMS system reached eighty percent of the global tpipe limit that is defined by the MAXTP parameter in the DFSOTMA system client descriptor.
9	X'09'	1	TMAMRSIM_WRNFLG2	Reserved
10	X'0A'	1	TMAMRSIM_WRNFLG3	Reserved
11	X'0B'	1	TMAMRSIM_WRNFLG4	Flags for resources in the fourth group of degraded resources. **X'08' - TMAMRSIM_W5MTP** The number of tpipes for this OTMA client has reached the maximum allowable number of tpipes set for this client on the MAXTP parameter of the OTMA client descriptor. No new tpipes can be created for this OTMA client until the number of tpipes drops. **X'04' - TMAMRSIM_W4MTP** The number of tpipes for this OTMA client has reached 80% of the maximum allowable number of tpipes set for this client on the MAXTP parameter of the OTMA client descriptor. **X'02' - TMAMRSIM_AWE** Message AWE reaches 80% flood **X'01' - TMAMRSIM_W4FLOOD** Flood warning for this client only. The number of OTMA messages waiting to be processed on the server is at eighty percent of the maximum allowable number defined for the server.
12	X'0C'	1	TMAMRSIM_NRSFLGS	Other flags for non-resource related indicators. **X'80' - TMAMRSIM_HB60S** Identifies this message as a heartbeat message. The server is available and resource usage is within normal limits. Heartbeat messages are sent every 60 seconds.

Table 164. Server state protocol command format (continued)

Dec offset	Hex offset	Length	Field name	Value and description
13	X'0D'	3	Reserved	
16	X'10'	16	TMAMRSIM_SRVNAME	The 16 character z/OS cross-system coupling facility (XCF) member name of the OTMA server.
32	X'20'	16	TMAMRSIM_CLTNAME	The 16 character XCF member name of the OTMA client.
48	X'30'	20	Reserved	
68	X'44'	12	TMAMRSIM_UTC	UTC time for this message

Related concepts

"Monitoring system resources with OTMA" on page 799
OTMA monitors IMS system resources that are used to process OTMA transactions and notifies OTMA clients about how well the IMS system is processing OTMA transactions.

Related reference

"Message-control information section" on page 859
For every OTMA message, you must provide message-control information in the first section of the OTMA message prefix.

Resume output for Tpipe

The state data for resume output for Tpipe portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

This command is sent by the client to force any queued output to be resent again. The number of tpipes and tpipe names are needed in the command.

If the hold queue for the tpipe exists and holds messages, those messages will also be sent to the client. The following table summarizes the format of state data for Resume Output for tpipe. The summary includes byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 165. Resume output for Tpipes command format

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section.
2	2	**Tpipe count**	Number of tpipe names in the command.
4	8	**Tpipe name**	The transaction tpipe name. Different tpipe names can be added as necessary.

Resume output for all Tpipes protocol command format

The format of the resume output for all Tpipes protocol command is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

The resume output for all Tpipes protocol command is sent by an OTMA client to request that OTMA resume sending queued output messages from all tpipes associated with the client. This protocol command does not resume output from tpipe hold queues.

For IMS to IMS TCP/IP communications, IMS Connect can issue this protocol command to resume output after a connection is restored to a remote IMS Connect instance. In this case, IMS Connect can include the name of the remote IMS Connect instance with this command and OTMA sends the oldest message on the tpipe if the message also includes the name of the remote IMS Connect instance.

The following table shows the format of the resume output for all tpipes protocol command in the state data section of the OTMA message prefix.

Table 166. Resume output for all Tpipes command format

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section.
2	2	**Reserved**	Reserved field.
4	8	**Name of a remote IMS Connect instance**	IMS Connect uses this field to retrieve messages on the tpipe queue that are destined for a specific remote IMS Connect instance.

Resume output for the hold queue for tpipe

The state data for resume output for the hold queue for tpipe portion of the OTMA message prefix is mapped by the TMAMHDR DSECT of the DFSYMSG macro.

An OTMA client sends the command to inform IMS to deliver one or all queued messages on the hold queue for tpipe. If this command is not issued, messages are held in the hold queue. However, the option that is specified in the command can be used to request how IMS holds and delivers messages. One of the four options in the following table can be specified in the State Data. If the client or z/OS cross-system coupling facility returns a NAK message to IMS, the current option is reset to `No-Auto`, which is the default.

The following table summarizes the format of state data for Resume Output for the hold queue tpipe. The summary includes, as appropriate, byte, length, content, hexadecimal value, and the description.

Table 167. Resume output for the hold queue for tpipes command format

Byte	Length	Content	Value	Description
0	2	**Length**		Length of the state-data section.
2	1	**Delivery option**	X'00'	*No-Auto:* exhaust all the messages in the queue only when the command is issued. This is the default.
			X'01'	*One Only:* deliver one message in the queue when the command is issued.
			X'02'	*Auto:* exhaust all the messages in the queue. After that, automatically deliver messages when they are queued.
			X'04'	*Auto-One:* deliver one message automatically when a message is available in the queue. The message might already be in the queue or it might be delivered later. After the message is delivered, this option is reset to `No-Auto`.

Table 167. Resume output for the hold queue for tpipes command format (continued)

Byte	Length	Content	Value	Description
3	1	**Callout mode**	X'80'	RESUME TPIPE call retrieves only synchronous callout messages.
			X'40'	RESUME TPIPE call retrieves both synchronous callout messages and asynchronous messages.
			X'20'	RESUME TPIPE call supports control data.
			X'10'	RESUME TPIPE call supports network security credentials.
4	8	**Resume tpipe token**		OTMA clients, such as IMS Connect, generate a resume tpipe token to uniquely identify RESUME TPIPE requests. When a tpipe supports multiple active RESUME TPIPE requests (MULTIRTP=Y) from IMS Connect, you can display the resume tpipe token with its associated alternate client ID by issuing the **/DISPLAY TMEMBER**(*tmemname*) TPIPE(*tpipename*) command. The token is also used when the client cancels a resume tpipe request.

Cancel resume output for tpipe hold queue request

The OTMA cancel resume output for tpipe hold queue request protocol command is sent by the client to cancel a resume tpipe request that was previously submitted by the client.

Upon receiving the request to cancel a resume tpipe, OTMA uses the resume tpipe token to locate the request to be canceled. If found, the resume tpipe request is discarded.

Table 168. Format of the cancel resume output for tpipe hold queue request protocol command

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section.
2	2	**Reserved**	Reserved field
4	8	**Token**	The resume tpipe token of the resume tpipe request to be canceled.

No messages on tpipe hold queue

The OTMA server sends the no messages on tpipe hold queue protocol command to inform the OTMA client that the tpipe hold queue does not contain a message or response for the current resume tpipe request.

OTMA issues this protocol command in when any of the following events occur:

- When the option of the "Resume Output for the Special Queue for Tpipe" is TMAMCRHQ_ONE and there is no IMS message for the client.
- When the option of the "Resume Output for the Special Queue for Tpipe" is TMAMCRHQ_NOAUTO and there is no IMS message for the client.
- When the option of the "Resume Output for the Special Queue for Tpipe" is TMAMCRHQ_NOAUTO and OTMA flushes all the existing messages in OTMA queue.

Table 169. Format of no messages on tpipe hold queue protocol command

Byte	Length	Content	Description
0	2	**Length**	Length of the state-data section.
2	2	**Reserved**	Reserved field
4	8	**Token**	The resume tpipe token of the resume tpipe request for which there are no messages.

Security data section

The security-data section is mandatory for every transaction or command, and is optional for OTMA protocol commands.

The security data portion of the OTMA message prefix is mapped by the TMAMSEC DSECT of the DFSYMSG macro.

The following table is a summary of the content of the security-data section of the message prefix. The summary includes, as appropriate, byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 170. Content of security data fields

Byte	Length	Content	Value	Description
0	2	**Length**		Length of the security-data section, including the length field.
2	1	**Security flag**	N	*No Security:* no RACF checking is done. It is assumed that the user ID and password are already verified.
			C	*Check:* RACF checks transactions and commands. Transaction and command authorization RACCHECKs are performed (TCLASS and CCLASS).
			F	*Full:* RACF checks transactions, commands, and regions. Transaction, IMS command, and MPP region authorization RACCHECKs are performed.
3	1	**Reserved**		
	1	**Utoken length**		Length of Utoken plus the length of Utoken Type. Length does not include length field itself.
	1	**Utoken type**	X'00'	Type of data to follow.
	*	**Utoken**		The user token. Variable length, from 1 to 80 bytes.
	1	**User ID length**		Length of the user ID plus the length of the User ID Type. Length does not include length field itself.
	1	**User ID type**	X'02'	Type of data to follow.

Table 170. Content of security data fields (continued)

Byte	Length	Content	Value	Description
	*	**User ID**		The user ID. Variable length, from 1 to 8 bytes.
	u	**Profile length**		Length of the profile plus the length of the Profile Type. Length does not include length field itself.
	1	**Profile type**	X'03'	Type of data to follow.
	*	**Profile**		The SAF profile. Variable length, from 1 to 8 bytes.
	1	**Network user ID Length**		Length of the network user ID plus 1 byte length of the network user ID type. The length does not include this length field itself.
	1	**Network user ID Type**	X'04'	Type of data to follow.
	*	**Network user ID**		Distributed user ID, which can be up to 246 bytes. For customers using IMS TM Resource Adapter, it is a Distinguish Name (DN) in the X.500 series of standards.
	1	**Network session ID Length**		Length of the network session ID plus 1 byte length of the network session ID type. The length does not include this length field itself.
	1	**Network session ID Type**	X'05'	Type of data to follow.
	*	**Network session ID**		Network session ID for the distributed user. Variable length from 1 to 254 bytes. For customers using IMS TM Resource Adapter, it is a domain name, realm, or registry name.

Related reference

“Explanation of OTMA security data fields” on page 884
The following information provides additional detail on the content of the security-data section of the message prefix.

Explanation of OTMA security data fields

The following information provides additional detail on the content of the security-data section of the message prefix.

Length
Specifies the length of the security-data section of the message prefix, including the length field.

Security Flag
Specifies the type of security checking to be performed. It is assumed that the user ID and password are already verified.

No Security
Specifies that no security checking is to be done.

Check
Specifies that transaction and command security checking is to be performed.

Full
Specifies that transaction, command, and MPP region security checking is to be performed.

Reserved
After the reserved field, the following three fields can be omitted or appear in any order. Each field has the following structure:

- Length field
- Field type
- Data field

The length field is not calculated in the length calculation. The actual length of the user ID or profile should not be less than the value specified for the length of each field.

Utoken Length
Specifies the length of the user token plus the length of the user token type.

Utoken Type
Specifies that this field contains a user token.

Utoken
Specifies the user token. The user ID and profile are used to create the user token. The user token is passed along to the IMS dependent region.

If the client has already called RACF, it should pass the Utoken with field type X'00' so that RACF is not called again.

User ID Length
Specifies the length of the User ID plus the User ID type.

User ID Type
Specifies that this field contains a user ID.

User ID
Specifies the actual user ID.

Profile Length
Specifies the length of the profile plus the length of the profile type.

Profile Type
Specifies that this field contains a profile.

Profile
Specifies the system authorization facility (SAF) profile. For RACF, this is the group name.

Network User ID Length
Specifies the length of the network user ID plus 1 byte length of the network user ID type. The length does not include this length field itself.

Network User ID Type
Specifies X'04' to indicate that the following data is the network user ID.

Network User ID
Specifies the distributed user ID, which can be up to 246 bytes. For customers using IMS TM Resource Adapter, it is a Distinguish Name (DN) in the X.500 series of standards.

Network Session ID length
Specifies the length of the network session ID plus 1 byte length of the network session ID type. The length does not include this length field itself.

Network Session ID Type
Specifies X'05' to indicate that the following data is the network session ID.

Network Session ID
Specifies the network session ID for the distributed user. It can be up to 254 bytes. For customers using IMS TM Resource Adapter, it is a domain name, realm, or registry name.

User data section

The user-data section of the OTMA message prefix is variable length and follows the security-data section. It can contain any data.

The user data portion of the OTMA message prefix is mapped by the TMAMUSR DSECT of the DFSYMSG macro.

The following table is a summary of the content of the user-data section of the message prefix. The summary includes, as appropriate, byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 171. Content of user data fields

Byte	Length	Content	Description
0	2	**Length**	Length of the user-data section, including the length field.
2	*	**User data**	The user data. Optional; variable length.

Related reference

"Explanation of OTMA user data fields" on page 886

The following information provides additional detail on the content of the user-data section of the message prefix.

Explanation of OTMA user data fields

The following information provides additional detail on the content of the user-data section of the message prefix.

Length
Specifies the length of the user-data section of the message prefix, including the length field. The maximum length of the user data is 1024 bytes.

User Data
Specifies the optional user data. This data is managed by the client, and can be created and updated using the DFSYDRU0 exit routine. The server returns this section unchanged to the client as the first segment of any output messages.

Reroute Tpipe
Specifies the reroute tpipe name for an input transaction submitted from an OTMA hold queue capable client, such as IMS Connect. This optional field starts at the offset X'5C' from the beginning of user data prefix and is used in the following two scenarios:

- When OTMA cannot deliver a CM0 IOPCB output message to the client, OTMA uses the reroute tpipe name, if specified, to reroute the output.
- When a send-only input transaction specifies the reroute tpipe name, any IOPCB output message which is the result of the Send-Only transaction will be queued to the reroute tpipe.

Application data section

The application-data section of OTMA messages is variable length and follows the user-data section of the message prefix.

You include IMS commands and transactions in the application-data section. The data in this section is unchanged by the receiver (server or client), and is transmitted directly to the server application program or to the client application program.

The application data portion of the OTMA message prefix is mapped by the TMAMAPP DSECT of the DFSYMSG macro.

The following table is a summary of the content of the application-data section of the message prefix. The summary includes, as appropriate, byte, length, content, hexadecimal value, the meaning, and includes usage comments.

Table 172. Application data

Byte	Length	Content	Description
0	2	**Length**	Length of the application-data section. The length includes the length field itself. The maximum length is 32 KB (32767 bytes).
2	2	**ZZ**	Application data IMS ZZ fields.
4	*	**Application data**	The optional application data. Multiple send requests might be required for a server output segment. For a client's transaction, the transaction code is specified in the first 8 bytes of the data area following the LLZZ. For transactions specified with MULTSEG, the standard IMS LLZZ format is required for each segment. The transaction code is only required in the first segment. Variable length. The maximum length of application data is 32 KB-4.

Sample OTMA messages

The following three sample OTMA messages are intended to show what OTMA messages look like when fully constructed, including the parts of the message prefix. The examples are not necessarily related to each other.

OTMA client-bid message

The following figure shows an OTMA client-bid message. The total length of the state-data section plus the security-data section of the message prefix is X'8C' bytes.

```
MESSAGE CONTROL INFORMATION:

   01102000 04004040 40404040 4040A0C0     |......        ff{|
   00000000 00000000 00000000 00000400     |................|

STATE DATA + SECURITY DATA:

   0036C3D3 C9C5D5E3 F1404040 40404040     |..CLIENT1       |
   40400100 00010003 00020100 00010003     |  ..............|
   0001C4C6 E2E8C4D9 E4F02000 00007FFF     |..DFSYDRU0....".|
   FFFF0000 00650056 C3525100 50018059     |.....e.V.RQ.P..Y|
   15569555 55555555 55555555 B7B686B0     |..n.........%&f[|
   81A61515 1B1B1B1B 1B1B1B1B B7B686B0     |aw..........%&f[|
   81A61515 55555555 55555555 8C918CA4     |aw..........j_u|
   15151515 55555555 55555555 09151515     |................|
   15151515 15151515 15151515              |................|
```

OTMA transaction message

The following figure shows an OTMA transaction message. The total length of the state-data, security-data, and application-data sections of the message prefix is X'D6' bytes.

```
MESSAGE CONTROL INFORMATION:

   01402000 0000E3D7 C9D7C5F1 4040A0D0     |. ....TPIPE1  ff}|
   00000001 00000000 00000001 00010000     |................|

STATE DATA + SECURITY DATA + APPLICATION DATA:
```

```
   00480020 0100E3C5 E2E3D4C1 D7400000     |......TESTMAP ..|
   00000000 00000000 00000000 0000C9D4     |..............IM|
   E2F0F0F0 F0F10000 00000000 00004040     |S00001........  |
   40404040 40404040 40404040 40404040     |                |
   40404040 40400000 0056C652 51005001     |      ....F...&.|
   80465551 95555555 55555555 55555555     |....n...........|
   55555555 55555555 55555555 55555555     |................|
   55555555 55555555 55555555 555586A3     |..............ft|
   A781B0B7 A4155555 55555555 5555B1B7     |xa[%u.........&%|
   8CB6A5A5 A415B7BD B7A41515 15150038     |%&vvu.%"%u......|
   0000C1D7 D6D3F1F8 4040E2C1 E840C8C5     |..APOL18  SAY HE|
   D3D3D640 40404040 40404040 40404040     |LLO             |
   40404040 40404040 40404040 40404040     |                |
   40404040 4040                           |                |
```

OTMA response message

The following figure shows an OTMA response message. The total length of the state-data, security-data, and application-data sections of the message prefix is X'EE' bytes.

```
MESSAGE CONTROL INFORMATION:

   01A08000 0000E3D7 C9D7C5F1 404080D0     |.ff....TPIPE1  .}|
   00000001 00000000 00000001 00010000     |................|

STATE DATA + SECURITY DATA + APPLICATION DATA:

   00480020 0100E3C5 E2E3D4C1 D740AB7F     |......TESTMAP %"|
   28EB9FAD 9A024040 40404040 4040C9D4     |.. Y..        IM|
   E2F0F0F0 F0F10000 00000000 00004040     |S00001........  |
   40404040 40404040 40404040 40404040     |                |
   40404040 40400000 0056C652 51005001     |      ....F...&.|
   80465551 95555555 55555555 55555555     |....n...........|
   55555555 55555555 55555555 55555555     |................|
   55555555 55555555 55555555 555586A3     |..............ft|
   A781B0B7 A4155555 55555555 5555B1B7     |xa[%u.........&%|
   8CB6A5A5 A415B7BD B7A41515 15150050     |%&vvu.%"%u.....&|
   0300D6E4 E3E2C5C7 40D5D67E F0F0F0F0     |..OUTSEG NO=0000|
   F140E2D7 C5C3C9C6 C9C5C440 E2C5C7E2     |1 SPECIFIED SEGS|
   C9E9C57E F0F0F0F8 F06B40E2 C5C7D5D6     |IZE=00080, SEGNO|
   7EF0F0F0 F0F34040 40404040 40F67EC3     |=00003       6=C|
   D6D340F6 F04040F7 7EC3D6D3 40F7         |OL 60  7=COL 7..|
```

Chapter 47. OTMA Callable Interface

The IMS OTMA Callable Interface (C/I) provides a high-level interface that allows application programs on other z/OS subsystems access IMS applications through OTMA.

The OTMA C/I API consists of API calls that are available to a C/C++ program. The API calls are used to join the IMS/OTMA z/OS cross-system coupling facility (XCF) group, to connect to IMS, to allocate communication sessions, to send IMS transactions/commands, to receive output from IMS, to close communication sessions, and to leave the XCF group.

The OTMA C/I API calls and sample OTMA C/I application programs are documented in *IMS Version 15 System Programming APIs*. Codes returned by the OTMA C/I are documented in *IMS Version 15 Messages and Codes, Volume 4: IMS Component Codes*

The following figure provides an overview of the OTMA C/I. Shown from left to right, in a sample z/OS environment, are sample C and C++ API calls (for example, OTMA_OPEN). The API calls pass through an object stub, the SVC interface routine, the API (for example, DFSYOPEN), and finally through the XCF group to IMS OTMA.

© Copyright IBM Corp. 1974, 2020

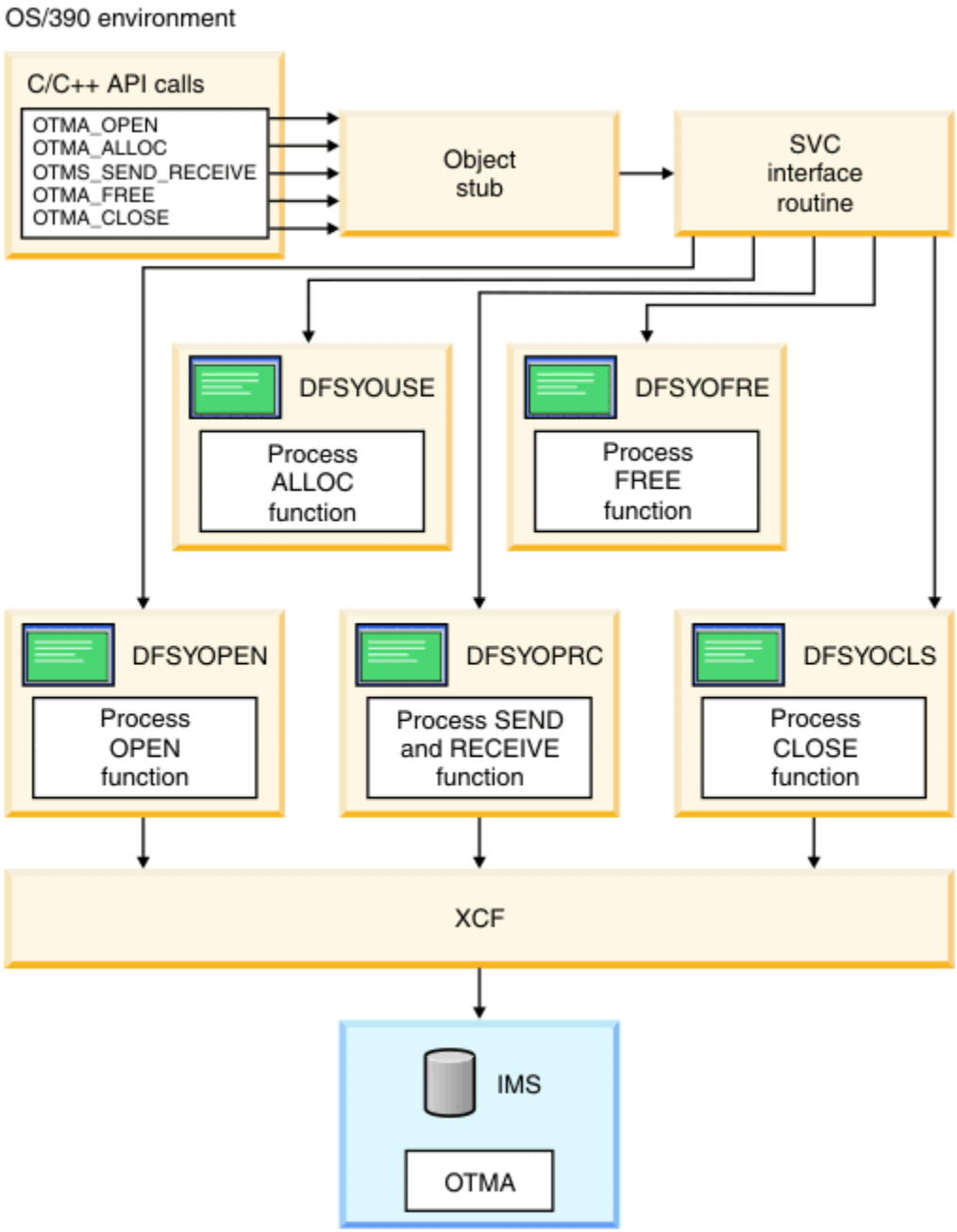


Figure 157. OTMA Callable Interface overview

The program that invokes the API calls can be running from an authorized or unauthorized library in problem or supervisor state. DFSYC0, a C header file, is provided to define the API calls. DFSYCRET, a load module, contains the entry points for each API call and is linked with the application program. OTMA C/I uses BPE SVC Services to process the API call.

Benefits: A key benefit of OTMA C/I is that it is easy to use.

Other reasons to use OTMA C/I are that it:

- Extracts out the details of OTMA and XCF
- Submits IMS transactions and commands
- Enables programs running from other z/OS subsystems to connect to multiple IMS systems
- Calls the APIs from an authorized or unauthorized library
- Connects to all IMS OTMA releases

OTMA C/I initialization

OTMA C/I provides a stand-alone program, DFSYSVI0, that must be run to initialize the OTMA C/I after either an IPL of z/OS or the application of maintenance to the OTMA C/I.

Note: Do not run the DFSYSVI0 program if any OTMA C/I clients are active.

If any OTMA C/I clients are active when DFSYSVI0 starts, DFSYSVI0 issues DFS3948E and DFS3950A. Running the SVC initialization when an OTMA C/I client is still active can result in XCF errors.

Active clients can be identified by issuing the z/OS command **D GRS,RES=(DFSOTMA,*)**.

If any clients are active when the DFSYSVI0 program is run, terminate the clients and reply "RETRY" to DFS3950A to attempt to initialize the SVC again.

If the clients cannot be terminated, reply "CANCEL" to DFS3905A to stop the utility without initializing the SVC.

If the clients are orphaned and will not be used again, then reply "BYPASS" to DFS3905A to proceed with the SVC initialization. Extreme care must be taken with this option as XCF errors may result if those existing clients attempt to use OTMA C/I again.

DFSYSVI0 invokes DFSYSVC0, one of the OTMA C/I modules. DFSYSVC0 loads and registers the SVC services by an authorized address space running on the same z/OS image as the application programs accessing it. Initializing the OTMA C/I after applying maintenance to the OTMA C/I ensures that when an OTMA C/I call is executed by an OTMA C/I client, all the OTMA C/I modules in the DFSYSVC0 load module that are needed to process the call are at the same level as DFSYSVC0.

Because there is only one OTMA C/I and OTMA C/I runs in the client address space through SVC rather than in IMS, DFSYSVI0 refreshes OTMA C/I for all OTMA C/I clients in the system regardless of which IMS systems the clients connect to.

You must add an entry in the z/OS program properties table (PPT) for the OTMA Callable Interface initialization program. The steps for doing this are:

1. Edit the SCHED*xx* member of the SYS1.PARMLIB data set.
2. Add the following entry to the SCHED*xx* member:

```
PPT PGMNAME(DFSYSVI0)   /* PROGRAM NAME = DFSYSVI0             */
             CANCEL     /* PROGRAM CAN BE CANCELED             */
             KEY(7)     /* PROTECT KEY ASSIGNED IS 7           */
             SWAP       /* PROGRAM IS SWAPPABLE                */
             NOPRIV     /* PROGRAM IS NOT PRIVILEGED           */
             DSI        /* REQUIRES DATA SET INTEGRITY         */
             PASS       /* CANNOT BYPASS PASSWORD PROTECTION   */
             SYST       /* PROGRAM IS A SYSTEM TASK            */
             AFF(NONE)  /* NO CPU AFFINITY                     */
```

3. Take one of the following actions to make the SCHED*xx* changes effective:
   - Re-IPL the z/OS system.

     or
   - Issue the MVS SET SCH= command.

   Related Reading: For additional reading about updating the program properties table, see *z/OS MVS Initialization and Tuning Reference*.

A sample JCL procedure for running DFSYSVI0 is as follows:

```
//OTMAINIT PROC RGN=3000K,SOUT=A
//*
//IEFPROC  EXEC PGM=DFSYSVI0,
//              REGION=&RGN
//*
//STEPLIB  DD   DISP=SHR,UNIT=SYSDA,
//              DSN=IMSVS.SDFSRESL
//*
```

```
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
```

After OTMA C/I is installed and initialized, you can use it to connect to any IMS OTMA release.

OTMA C/I security

To protect z/OS cross-system coupling facility groups from any non-authorized caller use `IMSXCF.OTMACI`, a RACF resource, defined in the RACF facility class for the OTMA C/I.

When the RACF resource is defined, `RACF RACHECK` is invoked before OTMA C/I performs a XCF `JOIN`. This method protects the access to XCF, the XCF group, and the member. This RACF checking is performed only when a non-authorized caller is using OTMA C/I.

OTMA C/I restrictions

Certain restrictions apply to the OTMA Callable Interface (C/I).

These restrictions include:

- C/I must be installed in a z/OS environment before it can be invoked. If C/I is not installed and invoked, an F92 abend occurs when otma_create or otma_open is issued. If C/I is not properly installed, a DFS3911E error message occurs.
- Application program languages other than C and C++ are not currently supported by OTMA C/I.
- All OTMA calls must be made in the same state (PSW key, supervisor or problem state, authorized or non-authorized) as the `otma_open` call. For example: If you were authorized when you did the `otma_open` call, you must be authorized for all subsequent calls.
- The resynchronization feature of IMS OTMA is not supported.
- IMS command `/SECURE OTMA PROFILE`, is not currently supported.

Timing out OTMA C/I sessions after otma_send_receive API calls for CM1 transactions

The OTMA C/I does not include a timeout function for the otma_send_receive API call and does not support the OTMA message-level transaction expiration function; however you can still implement a timeout function.

To implement a time out function for OTMA C/I client sessions use one of the following methods:

- Coding the EXPRTIME parameter on the TRANSACT system definition macro when you define the IMS transactions that will be invoked by the OTMA C/I client.
- Including a time out function with your OTMA C/I client.
- By both coding the EXPRTIME parameter and including a time out function in your OTMA C/I client.

Recommendation: Use the EXPRTIME parameter to implement a timeout function for transactions submitted through OTMA C/I instead of including a time out function with your OTMA C/I client. If the OTMA C/I client implements the time out function, although it is unlikely, a z/OS X'0C4' abend can occur if the OTMA C/I client releases the session storage after OTMA has received a response from IMS, but before OTMA posts the wait ECB.

If your OTMA C/I client includes a time out function, you can reduce the possibility of a z/OS X'0C4 abend by:

- When coding the OTMA C/I client time out function, include a delay in processing after the OTMA C/I client issues the otma_free call and before it frees the session storage.
- Specify a shorter time out value on the EXPRTIME parameter of the TRANSACT system definition macro for IMS transactions invoked by the OTMA C/I client.

Related reference
TRANSACT macro (System Definition)

Chapter 48. OTMA architected transaction attributes

When you issue an IMS **/DISPLAY TRANSACTION** command from OTMA, the output is in the form of an OTMA message, returned to the client in the application-data section of the message prefix.

The following table shows the Attributes Segment for a given transaction and describes the syntax of architected command output. The description includes byte, length, content, hexadecimal value, the meaning, and includes usage comments where appropriate.

Table 173. Transaction attributes segment

Byte	Length	Content	Value	Meaning	Comments
0	2	**Length**		Length of the Transaction Attributes Segment (LL).	This length includes the length field itself.
2	2	**ZZ**			
4	8	**Transaction code**		The 8-byte IMS transaction code.	
12	1	**Transaction type flag 1**			The flag 1 values are mutually exclusive.
		Valid	X'00'	A valid OTMA transaction.	
		CPIC	X'04'	A CPI-C (APPC) transaction.	
		FPX	X'08'	A Fast Path-exclusive transaction.	
		FPP	X'0C'	A Fast Path-potential transaction.	
		MSC	X'10'	An MSC remote transaction.	
		Invalid Syntax	X'FE'	Syntax error.	The data area contains the text of the error.
		Invalid	X'FF'	Transaction not found, or invalid.	
13	1	**Transaction type flag 2**			The flag 2 values are **not** mutually exclusive.
		Response	X'80'	An IMS response mode transaction.	
		Conversation	X'40'	An IMS conversational transaction.	
		Update	X'20'	The transaction has update capability.	
		Irrecoverable	X'10'	The transaction is defined as irrecoverable.	
		Multi-Segment	X'08'	The transaction has multiple segments.	
		Uppercase	X'04'	Uppercase translation requested.	
14	1	**Transaction status**			The indicated values are **not** mutually exclusive.
		STOP	X'80'	The transaction input queuing is stopped.	One of the following IMS commands was issued for the transaction: • `/STOP` • `/PURGE`
		OLC	X'40'	The transaction input queuing is stopped.	One of the following IMS commands was issued for the transaction: • `/MOD PREPARE` • `/MOD COMMIT`
		NOSCH	X'20'	IMS scheduling is stopped.	One of the following IMS commands was issued for the transaction: • `/PSTOP` • `/LOCK`
15	1	**Reserved**			
16	2	**Length**		Length (LL) of error text, if any.	If there is error text, it replaces all subsequent sections of the message.
18	*	**Error text**		Text of the error message.	Variable length. The error test section is applicable only if transaction type flag 1 is set to Invalid Syntax (X'FE')
16	8	**PSB name**		IMS PSB name.	Only present if there is no error text.
24	1	**Class**		SMB message class for IMS scheduling.	

© Copyright IBM Corp. 1974, 2020

Table 173. Transaction attributes segment (continued)

Byte	Length	Content	Value	Meaning	Comments
25	1	**Current priority**		Current SMB priority.	
26	1	**Normal priority**		Normal SMB priority.	
27	1	**Limit priority**		SMB limit priority.	
28	2	**Enqueue count**		Number of messages enqueued.	
30	2	**Dequeue count**		Number of messages dequeued.	
32	2	**Enqueue limit**		Enqueue limit count.	
34	2	**Processing limit count**		Processing limit count.	
36	2	**Output max segment length**		The maximum output segment length.	
38	2	**Output limit of message segments**		The output limit of message segments.	
40	2	**Parallel limit**		The PARLIM value from TRANSACTION statement.	
42	1	**Region count**		The number of regions in which the transaction is currently scheduled.	

Part 11. SLU P and Finance Communication

These topics discuss in detail SLU P and Finance Communication, including an overview of SLU P and Finance communication, the IMS facilities used for SLU P and Finance communication, network operation with SLU P and Finance communication, and SLU P network protocols.

© Copyright IBM Corp. 1974, 2020

Chapter 49. Overview of SLU P and Finance Communication

This topic and the topics below present an administrative overview of SLU P and Finance Communication Systems, and explains how IMS implements this architecture.

User-written application programs for IBM Finance Communication Systems can be defined to operate in two different ways with IMS—as an IBM 3600/4700 Finance Communication Controller or as a secondary logical unit type P (SLU P). These systems are described as UNITYPE=FINANCE and UNITYPE=SLUTYPEP, respectively, on the TYPE macro statement.

Definitions:

- The term *SLU P* is used in the following topics when describing support that is applicable to both system types.
- The terms *Finance* and *SLUTYPEP* are used in the following topics when it is necessary to distinguish between the systems.

A major difference between SLU P and Finance systems is the level of MFS support available to workstations in the programmable control unit.

- Finance—Workstations can be identified as displays, printers, passbook printers, and ATMs (automated teller machines), such as the 4730 systems. MFS can provide detailed support for display paging, printer page formatting, and ATMs.
- SLU P—Each LU in the controller has a program name and its unique device characteristics are unknown to IMS. The programmable controller is responsible for device control and formatting. The MFS administrator and the program administrator for the remote controller need to establish data field structures for exchanged messages. MFS accepts the input structure and rearranges it as required for the IMS application program. At output, MFS accepts application program-provided data and converts it to the correct format for transmission to the controller. This MFS facility is called distributed presentation management (DPM).

Related tasks

"ETO and 3600/Finance and SLU P" on page 105
You can sign on to static system-defined 3600/Finance and SLU P terminals in one of two ways: using the **/SIGN** command or using logon user data.

The IMS-SLU P network

The three major elements of an IMS-SLU P network are the controller and the terminals executing the remote program, the communication link, the central processor executing the host IMS system.

Each element includes programming to perform a part of the total data processing performed by the system. IMS resides in the host processor and communicates with an application program in the SLU P system's controller through the communication link. The controller's application program monitors the terminals attached to the controller.

IMS supports 4700 terminals attached to an IBM 4701/4702 Finance Communication Controller. IMS supports the IBM 4730 Personal Banking Machine, both directly attached to a 37x5 as a SLU P and attached to a 4701/4702 as part of the SLU P system.

Related reading: For a list of 4700 terminals, see *IBM 4700 Finance Communication System: System Summary*.

IMS also supports 3600 terminals attached to either an IBM 3602 or 4701/4702 as part of a Finance system. (IMS does not distinguish between 3600- and 4700-series terminals.)

Examples of other IBM products that connect to IMS with the SLU P protocols are the Series/1, the IBM 3650, and the IBM 8100.

© Copyright IBM Corp. 1974, 2020

Each of these terminals, or workstations, can be defined to IMS as a component of the appropriate SLU P system using the COMPT or ICOMPT keyword on the TERMINAL macro statement.

System configuration

Before configuring a SLU P system, you must identify the financial operations that are required of the planned system. After you have done this, you can define to IMS the configuration of the logical units and components.

The SLU P system is regarded by IMS as a subsystem and consists of one or more logical units.

Definition: When configuring the system, you can define one or more *logical units*. These logical units can consist of devices, storage, and application programs. These logical units are referred to as *logical workstations* in Finance Communication System publications.

SLU P and Finance workstations

A workstation can consist of one or more terminals. Each workstation performs a specific type of financial operation, such as teller operation (deposits and withdrawals), report printing, or cash dispensing.

The workstation, as defined to IMS, does not need to reflect the actual devices attached to it at the 4701/4702, but the workstation must reflect the view of the workstation as seen by the IMS system.

A sample configuration is available from the IMS library (IMS.ADFSMAC; member name=DFSCP360).

System controller application program

The operation of a workstation is controlled by a user-written application program that resides in the SLU P system's controller. The application program can be designed to control one or more workstations and needs only to perform the functions required by its workstations.

The functions available to the application program include:

- Reading and writing to the terminals associated with the workstation
- Editing and verifying the data received from the terminal
- Reading and writing to the controller diskette
- Reading and writing to the host processor
- Editing and verifying data received from the host processor
- Operating offline when the host processor or IMS is unavailable

Writing the controller application program with MFS and XRF

If your system uses MFS and XRF, there are certain actions you must take when writing a controller application program.

When writing the controller application program for a system that uses MFS, do each of the following:

Note: Do not specify the WL (warning line) option on the 4700 controller's TERMINAL macro instruction. Specifying the WL option can cause an unexpected Finance system data exception error to occur on an LWRITE instruction.

- Specify the PS (page size) option on the 4700 controller's TERMINAL macro only if EJECT is specified on the IMS MFS DEV statement.

  If PS is not specified, the EJECT from IMS results in a new line function. If the "end-of-page" condition occurs on a workstation, an unexpected Finance system data exception error can occur. An EJECT from IMS resets the line count, resulting in an "end-of-page " condition.

  If the CFOLD option is specified on the 4700 controller's TERMINAL macro instruction, the formats defined for MFS should not cause spacing for the center fold on a passbook. The SLU P system performs this function automatically.

- Check to see if the IMS MFS DFLD statement specifies ATTR=YES.

  If it does, any attempt to print an underscore (X'6D') can result in either of the following:

  – A data check on a 3610/3612

    Specify a print position for the EBCDIC value (X'6D') when the OUTRTBL statement specifies other than the 128-character set for the 3610/3612 device.

  – Printing of a blank on a 3618 device

    Specify a print position for the EBCDIC value (X'6D') when the OUTRTBL statement specifies the 48-character set for a 3618 device.

- Check to see if the MFS DEV statement option of EJECT, BGNMXG, BGNPP, or ENDPP is used to send eject characters (X'0C') within one IMS output transmission (other than the last character).

  If so, an "intervention-required" condition occurs on the passbook printer after the eject. In this situation, your application program in the controller must be capable of issuing multiple LWRITE instructions to the passbook printer to print the IMS output transmission. Each LWRITE should include the data up to and including an eject character, or up to the end of the transmission.

Related reading: For more information on coding the controller's macros, see *IBM 4700 Finance Communication System: System Summary*.

Considerations for controller application programs for XRF systems

If your SLU P system operates in an XRF complex, your controller application program must be able to handle messages lost during an XRF takeover.

Many messages that are "in-flight" when the alternate system takes control are recovered with no action from the application program or terminal operator. In this case, the takeover is transparent.

When the takeover is not transparent, message DFS3861I is issued in one of the following combinations:

- Message DFS3861I only—If the output message is recoverable, this indoubt message is issued. If irrecoverable, this message is not issued. If you are receiving output from IMS, and the new active IMS system cannot determine if you received the message, IMS requests an exception response but is not receiving one from your program.
- A CANCEL command followed by message DFS3861I—multi-segment output is in progress (last-in-chain not yet sent). If the in-flight message is recoverable, you should receive the in-flight message again. If the in-flight message is irrecoverable, the next in-flight message (if one is on the queue) is issued.
- An exception response followed by message DFS3861I—your input message is lost. Resend your last input message.

Converting controller application programs from Finance to SLU P

All current functions available for logical units defined to IMS as Finance, except the SCAN/NOSCAN option and Finance MFS, have similar functions for logical units defined as SLU P.

IMS does not scan output for the control character sequence when sending output to a secondary logical unit type P. Therefore, the implied option is NOSCAN. If an application program uses MFS, it must be converted to use the distributed presentation management (DPM) option, which divides responsibility for message formatting between MFS and a user program residing within the logical unit. With the use of the DPM option, physical terminal characteristics are not defined to MFS. MFS formats and presents data to the user program component of the SLUTYPEP. The user program must complete the formatting, if necessary, and present the data to a physical device.

To execute correctly when defined to IMS as SLU P, existing remote user-written programs that run in a Finance controller must be converted as follows:

- Any input message headers must be converted to SLU P message headers.
- Any input message indicating only begin-bracket must also indicate change-direction.

- If the TERMINAL macro SCAN option is specified, the Finance controller application program must be converted to detect and process the same options provided by the SCAN option.
- If the controller application program uses MFS, the remote application must be converted to use the device-independent MFS distributed presentation management option.
- When converting from Finance-specific MFS to DPM, the MFS format descriptions must be redefined and recompiled using the IMS MFS Language utility.

VTAM facilities used

The physical transmission of data between IMS and a SLU P system is controlled by VTAM.

Related reading: For more information on the communication facilities that govern data transmission between IMS (a VTAM application program) and the controller, see *Network Program Products General Information* and *z/OS Communications Server: SNA Programming*.

The VTAM facilities that IMS uses, particularly for a SLU P system, include:

- Connection, disconnection, and establishing logon mode
- Messages and responses
- Definite response 1 and definite response 2 (All data transmitted between IMS and a workstation must request a definite response 1, definite response 2, definite response 1 and 2, or exception definite response 1 or 2. VTAM commands must request definite response 1.)
- Sequencing and chaining
- Quiescing
- Facilities for ensuring orderly communication, including the use of brackets and change-direction indicators. In SNA protocol, a bracket is one or more chains of request units (RU) and the responses that are exchanged between two LU-to-LU half sessions that represent a transaction. In VTAM, a change-direction indicator means that the sender has finished sending and is prepared to receive.
- Sequence-number recovery
- Receiving input, sending messages
- Unconditional bracket termination
- Primary Error Recovery Procedure (ERP)

  The ERP is terminated by an IMS output error message sent to the workstation.

IMS also supports the class of service (COS) and session outage notification (SON) facilities.

Related concepts

"Using SON/COS support in IMS" on page 419
Session outage notification (SON) and class of service (COS) are facilities of VTAM and SNA that allow IMS to recognize a session outage.

Related reference

"Error handling" on page 948
The following topics describe the procedures used by IMS and required of the controller or the controller application program to handle failures resulting from transmission or protocol errors.

VTAM commands and indicators used with SLU P

VTAM commands and indicators (communication control information) are necessary for data transmission between an IMS application program and another VTAM logical unit.

The following table shows which VTAM commands and indicators IMS sends to and receives from the SLU P controller. Results are unpredictable if any unsupported commands or indicators are used.

With the exception of the request-recovery command and the change-direction indicator, which are described following the table, the commands and indicators operate as described in *z/OS Communications Server: SNA Programming*.

Recommendation: During a SLU P session, every input message must have either a change-direction (CD) or end-bracket (EB); otherwise, the session is terminated.

Table 174. VTAM commands and indicators sent and received by IMS

VTAM command/indicator	IMS sends to controller	IMS receives from controller
INDEPENDENT OF SESSION		
Initiate Session		X
Procedure Error	X	
Terminate Session		X
SESSION CONTROL		
Bind	X	
Clear	X	
Request-Recovery (RQR)		X
Set-and-Test-Sequence Numbers (STSN)	X	
Start Data Traffic (SDT)	X	
Unbind	X	
NORMAL (synchronous) FLOW		
Begin-Bracket (BB)	X	X
BID	X	
CANCEL	X	X
CHASE		X
End-Bracket (EB)	X	X
Logical Unit Status (LUS)		X
Quiesce Complete (QC)	X	
Ready-to-Receive (RTR)		X
Change Direction (CD)		X
EXPEDITED (asynchronous) FLOW		X
Quiesce-at-end-of-chain (QEC)		X
Release Quiesce (RELQ)		X
Request-Shutdown (RSHUTD)		X
Shutdown (SHUTD)	X	
Shutdown Complete (SHUTC)		X
Signal		X

Request-recovery command

Upon receipt of the VTAM Request-Recovery (RQR) command, IMS issues a CLEAR. The SLU P application programmer must exercise care when sending this command while a recoverable IMS output message is in progress.

To send an RQR command to IMS, the application program must first respond to the output operation that is in progress. If the application program does not send the reply but instead sends only the RQR command, the controller automatically sends the outstanding DR2 reply at the LEXIT or next LREAD statement. If IMS receives the RQR command before the DR2 reply, the CLEAR generated by IMS can cause the DR2 reply to be lost. The status of the current output message is then unpredictable.

Restriction: Session initiation and resynchronization caused by an SNA Request-Recovery (RQR) command is not allowed when the node is in response mode and the response reply message is not yet available for output; that is, the input response mode transaction is still queued or in the process of execution. Response-mode transactions are not recoverable or restartable prior to the application sync point; therefore, session input acknowledgment does not occur until input processing is complete.

Related concepts

"Message resynchronization" on page 918
The purpose of message resynchronization is to guarantee the integrity of messages across sessions.

Change-direction indicator

IMS supports use of the change-direction indicator on the last chain element of an input message or a VTAM command. This allows device support to be more consistent with VTAM operation. The operation of IMS is not affected by the indicator.

Establishing connection and specifying logon modes

When establishing connection using the VTAM OPNDST macro instruction, IMS specifies session parameters that define the rules that a logical unit must follow when communicating with IMS.

IMS examines either a user-supplied or a VTAM default set of session parameters, and overlays only those parameters on which IMS has dependencies. The remaining bytes are not changed. You can include user data with the BIND parameters.

User data included within the BIND is used by IMS as input to the signon process and is similar in function to the IMS **/SIGN** command. This is required for ETO Finance terminals and SLU P terminals, but is optional for static terminals.

If you do not specify a mode name on the TERMINAL macro statement during IMS system definition or on the logon descriptor, you can supply a mode name using any of the following methods:

- On the VTAM network operator **VARY** command
- On the IMS **OPNDST** command
- On a request for system initialization by another logical unit

If you specify a mode name on the IMS TERMINAL macro statement or on the logon descriptor, it is used, unless you override it using the IMS **/OPNDST** command or the VTAM **VARY** command.

If you do not supply a mode name during IMS system definition using a logon descriptor, on an IMS command, or using CINIT, VTAM assumes a default mode name.

Related reading: For more information on establishing logon mode tables and defining logon mode table entries, see *z/OS Communications Server: SNA Resource Definition Reference*.

You can override the use of the default logon mode table entry in one of two ways:

- At system definition, you can specify the logon mode entry on the TERMINAL macro using MODETBL= keyword.
- You can specify on the MODETBL keyword of the **/OPNDST** command that the logon mode table entry replace the table entry defined at system definition.

IMS overrides the session parameters for function management, transmission services, and primary and secondary network protocol.

Related reference

Bind parameters for SLU P and LU 6.1 (System Programming APIs)

TERMINAL macro (System Definition)

/OPNDST command (Commands)

Establishing connection with the XRF complex

To establish a session with IMS in an XRF complex, your logon request must include either the USERVAR name you defined in the USERVAR tables or the MNPS ACB name shared by both of the IMS systems in the XRF complex.

In the case of XRF with USERVAR, VTAM matches the USERVAR logon message to the IMS application that is currently active. For XRF with MNPS, VTAM links directly to the active IMS's MNPS ACB.

If you use XRF with USERVAR, SLU P terminal programs cannot use the APPLID name in the PLUNAME field of the BIND to reestablish a session with IMS. Instead, SLU P terminal programs must use the USERVAR in the user data field of the BIND. This restriction does not apply to XRF with MNPS, because this type of XRF system does not use a USERVAR.

SLU P systems are normally defined as class-1 terminals; in most cases in an XRF complex, when an alternate IMS takes over processing from an active IMS, the takeover is accomplished without losing class-1 terminal sessions. The terminals might or might not be aware of the takeover.

If you are connecting the 4730 Personal Banking Machine to an XRF IMS as a SLU P device, use the following machine data configuration options:

- Invalid message: 10 or 11
- Multiple commands: 10 or 11

The first digit (1) prevents the 4730 from sending an exception response to certain network messages, including the DFS3861 message sent after an XRF takeover. Exception responses to these messages terminate the SLU P session between the 4730 and IMS. The second digit (0 or 1) indicates whether errors should be logged.

Related reading: For more information on the 4730, see *IBM 4730 Personal Banking Machine Operations Support Manual*.

Related concepts

“Considerations for controller application programs for XRF systems” on page 901
If your SLU P system operates in an XRF complex, your controller application program must be able to handle messages lost during an XRF takeover.

XRF variations: MNPS and USERVAR (System Administration)

Specifying MNPS to manage terminal sessions for XRF (Operations and Automation)

Related reference

Finance communication system bind parameters (System Programming APIs)

Defining IMS.PROCLIB members for XRF (System Administration)

DFSHSBxx member of the IMS PROCLIB data set (System Definition)

Bracket and send/receive management

The change-direction and bracket indicators are used to begin, end, and control the direction of synchronous transmissions between a SLU P system and IMS.

IMS support of these indicators is summarized in the following table (X = supported).

Table 175. Use of VTAM bracket and change-direction indicators

Synchronous transmissions	BB	EB	BB/EB	BB/CD
Received by IMS with data	X		X	X
Sent by IMS with data		X	X	
Received by IMS with data flow synchronous commands (normal flow)		X		

IMS does not support the absence of bracket indicators or the use of end-bracket-only on inbound synchronous data. IMS does not send output synchronous data with begin-bracket (BB) only, change-direction (CD) only, or BB/CD. Because IMS always ends a bracket (unconditional bracket termination) on input synchronous data, it is unnecessary to send either EB or CD on output synchronous commands. Input synchronous commands can specify EB or CD, but must be consistent with the current bracket and send/receive states; otherwise, the session is terminated. IMS does not send VTAM indicators with data flow synchronous commands (normal flow).

Chapter 50. IMS facilities used for SLU P and Finance

The following topics describe in detail the IMS facilities that support the Systems Network Architecture (SNA) environment.

Component definition

IMS considers a *workstation* to be one physical terminal. If a workstation is made up of more than one device, each physical device that makes up the workstation must be defined as a component of that workstation.

Each component can have an IMS logical terminal (LTERM) associated with it. One LTERM is associated with each component. A user-written MPP can address specific components of a workstation through the appropriate LTERM.

IMS assumes that all input is from the first LTERM in the component list that passes the necessary operational and security checks. Message switches, broadcast messages for specific logical terminals, and data replies to transactions are directed to the component associated with the specified output LTERM.

LTERM naming

To facilitate the use of the CHNG call, define a convention for naming LTERMs. One method is to set the LTERM name to be a combination of the workstation and component identifiers.

Example: IMS considers the 4704 keyboard and display system components as one component, the 4706 magnetic stripe reader as another component, and the 4710 receipt printer as yet another component. Thus, workstation 100 can have three components: WS100DS (4704), WS100MS (4706), and WS100RP (4710). Such a standard permits an MPP to interrogate the I/O PCB (LTERM name field) to identify the workstation and then to specify the proper alternate PCB for output using the CHNG call.

For a multiple-component terminal, such as a SLU P, you must ensure that IMS creates enough queues to service the terminal. You can use an exit routine or a specific user descriptor for that unique logical unit to ensure that IMS creates the required number of queues.

Output component selection

IMS system definition allows a workstation to have a maximum of four output components.

If more than four output components are required for a workstation, or multiple LTERMs are not desired, a user-defined MPP-to-workstation protocol and data format must be provided.

Workstation components are assigned an identification number: X'01', X'02', X'03', and X'04'. For Finance terminals, the identification number is based on the order in which they are defined. For SLU P terminals, the identification number defaults to Comp[1], (Program1, Basic). IMS nonqueued system messages have no specific destination and are directed to the first component.

An IMS Transaction Input edit routine can be used to append the station's node name to the input message. The MPP can reference the node name to determine which LTERM name to use for output. For terminals created using the Extended Terminal Option (ETO) feature, append the user name to the input message because the user could be signed off.

To return a reply message to a workstation operating in terminal response mode, or to a conversational transaction, the MPP must insert the reply to the I/O PCB. However, for some stations, that reply message might be intended for a component other than the one associated with the LTERM specified in the I/O PCB. Because the destination of the I/O PCB cannot be modified by the CHNG call, an alternate PCB type is required. The alternate PCB type must fulfill these response requirements and also be modifiable. The PCB type, called the response alternate PCB, can be defined using PSBGEN. This PCB can be used instead of the I/O PCB to return reply messages to conversational transactions and to stations operating

© Copyright IBM Corp. 1974, 2020

in terminal response mode. It can also be defined as modifiable. This allows the CHNG call to be used with this PCB to select the appropriate destination for the reply. When the CHNG call is used, the LTERM specified in the call must be assigned to the same physical terminal as the LTERM specified in the I/O PCB. If this is not the case, a status code is returned on the ISRT call.

Related tasks

"Administering the Extended Terminal Option" on page 69
The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

Input component determination

Proper relationships between input and output components can be established using the NAME macro during IMS system definition, or by using an ETO user descriptor and the Signon exit routine.

This relationship allows the terminal to specify its input component and causes output to be returned to a component that is defined during IMS system definition. Proper definition and use of input components can reduce or eliminate the need for LTERM naming conventions, MPP change calls, and inserts to alternate PCBs.

A user can establish a connection for SLU P usage by defining a component for each of the operators, devices, or processing requirements controlled by a remote application program.

IMS performs input component selection for the remote logical units and assumes that all input is from the first LTERM in the list that passes the necessary operational and security checks. For input from SLU P, input component selection is performed based on the component indicated in an optional input message header. If no function management header is received, IMS assumes the input is to be associated with the LTERM for component 1. Message switches, broadcast messages for specific LTERMs, and data replies to transactions are directed to the component associated with the specified output LTERM.

Terminal-response mode

Terminal-response mode is a mode of operation that can be defined for transactions, users, and terminals attached to IMS.

When a SLU P station is operating in terminal-response mode, all operations are stopped between the workstation and IMS from the time IMS receives a transaction until IMS receives acknowledgment that the reply message has been received by the workstation. For normal IMS output messages, this acknowledgment is the receipt of the DR2 response requested for recoverable transaction output. For Fast Path output messages, this acknowledgment is the receipt of the next input message or an RTR command. For MFS-paged output messages, this acknowledgment is the receipt of the next input message, MFS NEXTMSG or NEXTMSGP control commands, or a requested DR2 response (if not Fast Path output) on the last page of the message. Output caused by a nonrecoverable transaction requests an exception DR2 response and does not require a response from the workstation. Terminal-response mode can reduce the processing required by the controller application program.

Terminal-response mode can be selected by an installation during IMS system definition or with an ETO user descriptor. Your system can be defined as forced, negated, or transaction-dependent:

- If it is defined as forced, every input transaction is in terminal-response mode.
- If it is defined as negated, no input transaction is in terminal-response mode.
- If it is defined as transaction-dependent, terminal response mode is determined on a transaction-by-transaction basis. The use of transaction-dependent terminal-response mode can make the controller application programs more complex, because they must handle communication protocols resulting from both forced and negated terminal-response mode environments.

Terminal-response mode can only be invoked by valid transactions, not by message switches, IMS commands, VTAM commands and indicators, or MFS control requests. Transactions that are unacceptable (for example, because of a security violation or an invalid transaction code) cannot invoke terminal-response mode.

When a workstation is operating in terminal-response mode, the following processing occurs after the workstation has established a session with IMS:

1. The workstation sends an input transaction.
2. IMS places the workstation in terminal-response mode.
3. IMS passes the transaction to a message processing program (MPP).
4. The MPP processes the transaction.
5. The MPP returns a reply, using either the I/O PCB or an response alternate PCB.
6. IMS returns a DRx response, if one is requested.
7. IMS sends the reply to the workstation.
8. The workstation returns a DR2 response, if one is requested.
9. IMS removes the workstation from terminal-response mode.

While a workstation is in terminal-response mode, IMS accepts no input from it, and sends no output to it other than the reply from the MPP. If the MPP abends while processing the input transaction, IMS might send an exception response and the associated IMS error message. Any output that is not in reply to the transaction that initiated terminal-response mode is held in IMS's output queue. After IMS removes the workstation from terminal-response mode, it sends the unsolicited output.

When Fast Path is used, the following processing occurs:

1. The workstation sends an input transaction.
2. IMS places the workstation in terminal-response mode.
3. IMS passes the transaction to a Fast Path message processing program (IFP).
4. The IFP processes the transaction.
5. The IFP returns a reply, using either the I/O PCB or a response alternate PCB.
6. IMS returns DRx, if one is requested.
7. IMS sends the reply to the workstation.
8. An exception DR2 response is required on output; therefore, the workstation sends the next input message or RTR command. If IMS has output for the workstation on the message queue, a definite response is required. The workstation sends DR2.
9. IMS removes the workstation from terminal-response mode.

The master terminal operator can reset a workstation in Fast Path terminal-response mode before the response is returned by issuing the **/STOP NODE** and **/START NODE** commands in sequence from the master terminal.

Defining a workstation for terminal-response mode

You must understand the operation sequences of terminal response mode before defining a workstation to operate in terminal-response mode.

Less processor time is required for stations that operate in terminal response mode, and the controller application program that controls such a workstation might also be less complex.

The following considerations also apply:

- Typical data collection applications cannot be performed from stations that operate in terminal-response mode. In this mode, a reply from the MPP is required for each transaction before IMS accepts another transaction. Waiting for this response extends the time of the data entry process.
- IMS does not send a response to an input message until the output reply is available. Thus, when the response is returned, it indicates that the input has been processed and the application has reached a sync point.
- While a workstation is in terminal-response mode, IMS does not attempt to obtain any more input from that station. Master terminal operator intervention is required if an error prevents creation or

transmission of a reply. Some conditions can prevent a reply being sent to the workstation. These include:

  - The LTERM stopped.
  - IMS was unable to schedule an MPP (database stopped, MPP stopped).
  - An MPP logic error caused no reply to be returned except for EMH (expedited message handler), which generates a zero-length reply.

  If any of these conditions occurs, the workstation is temporarily inoperative. Before the workstation can be used again, the master terminal operator must diagnose and correct the error.

- A response message remaining on the IMS output queue or inserted by the user-MPP after session termination is re-sent after initiation of the next session. If the BID option is specified, this message is preceded by the VTAM BID command. Both the begin-bracket and end-bracket indicators are sent with the message.

System analysts must evaluate these factors for their own system application programs and operating environment.

Related concepts

"Terminal-response mode" on page 908
Terminal-response mode is a mode of operation that can be defined for transactions, users, and terminals attached to IMS.

Output messages sent while in a between-brackets state

If an IMS output message is sent while the workstation is not protected from output and in a between-brackets state, IMS sends one of two things.

IMS sends either:

- The message with both the begin- and end-bracket (that is, bids for bracket with data) if the workstation is defined with the NOBID option
- The BID command to request permission to begin a bracket if the workstation is defined with the BID option.

A workstation can accept the bracket with any requested DR1 or DR2. A DR1 to the bid causes IMS to send the output message with both begin- and end-brackets.

A workstation can reject the bracket with an appropriate exception DR1 or DR2.

When a workstation is ready for output after rejecting a bracket, it sends the ready-to-receive (RTR) command to request output from IMS.

The following steps show how IMS handles output for a workstation that is in terminal-response mode and that is defined with the BID option when the session is between-brackets:

1. When IMS receives a message switch for a workstation defined with the BID option, IMS places the message in the output queue.
2. After removing the workstation from terminal-response mode, IMS sends a BID command to notify the workstation that output is pending while between-brackets. IMS requests a DR1 response to the BID command.
3. The workstation returns the DR1 response if it is ready to receive the output.
4. IMS sends the output when it receives the DR1.
5. The workstation returns a DR2 response, if requested.
6. If the workstation is not ready to receive the output, it returns an exception DR1 response indicating "bid rejected". If the output is recoverable, IMS returns the message to the queue and waits until the workstation indicates it is ready to accept the output. If the exception response indicates "RTR will not follow" and the output is nonrecoverable, IMS discards it. If the exception response indicates "RTR will follow," IMS returns the message to the queue (regardless of recoverability) and waits for a VTAM ready-to-receive (RTR) command.

7. When the workstation is ready to receive the output, the workstation sends the VTAM RTR command and requests a DR1 response.
8. If the output message that caused the bid is still available, IMS returns the DR1 response, followed by the output message.
9. The workstation returns a DR2 response (12).

While IMS waits for the RTR command from the workstation, the workstation can perform any type of processing desired. IMS accepts and responds to transactions during this time, but does not transmit any unsolicited output until it receives the RTR command.

If IMS no longer has output available to send when the workstation sends RTR (perhaps because of an intervening **/ASSIGN** or **/DEQUEUE** command), or if the data was irrecoverable and IMS discarded it, IMS returns an exception DR1 response and an error message indicating no output is available.

Related concepts

"Display screen protection for finance stations" on page 914
When a Finance station is defined with the NOBID option, IMS provides support similar to the screen protection function for the IBM 3270 Information Display System.

Related reference

"Controller or station-detected errors" on page 949
Whenever the controller detects an error on a message from IMS, or simply cannot accept the message at that time, an exception DR2 that includes 4 bytes of sense data is returned.

Designing for output messages sent while in between-brackets state

When IMS is between-brackets, it sends the BID command to bid the workstation before actually sending an output message.

If the bid is rejected, the recoverable output message remains queued if the exception sense data indicates an RTR is forthcoming; otherwise, the message is discarded.

If IMS receives an exception response to an irrecoverable output message (inquiry-only transaction), the message is lost, because IMS dequeues the message immediately upon sending it to VTAM.

Related concepts

"Output messages sent while in a between-brackets state" on page 910
If an IMS output message is sent while the workstation is not protected from output and in a between-brackets state, IMS sends one of two things.

Related reference

"Controller or station-detected errors" on page 949
Whenever the controller detects an error on a message from IMS, or simply cannot accept the message at that time, an exception DR2 that includes 4 bytes of sense data is returned.

IMS Message Format Service

The following topics describe some IMS MFS facilities that specifically apply to the SLU P system.

Related concepts

IMS Message Format Service (Application Programming)

MFS message formats (Application Programming APIs)

Designing MFS for the workstation environment

Both input and output data can be processed by MFS. The exit for the Input Transaction edit routine is available to provide additional editing capabilities.

The availability of MFS to workstations is defined on an individual basis by the system administrator during IMS system definition. When MFS is defined as available, each input or output message can

optionally be processed using MFS. The format of messages to be processed by MFS is defined using the IMS-supplied MFS Language utility.

Operations using MFS can be quite different from operations using the IMS Basic Edit facilities. Device formats and operator procedures should be designed carefully, with the objectives of easy use and high-operator productivity.

MFS provides two levels of message formatting: device-level message formatting and distributed presentation management (DPM). Device-level message formatting is for Finance logical units. DPM support is for SLU P logical units.

To prevent null screens, a null record (data length = 0) produced by MFS as the result of a user-specified MFS format description is not sent by IMS.

MID/MOD chaining

MFS input formatting for a workstation occurs when a message input descriptor (MID) name is provided with an input message.

The MFS message output descriptor (MOD) can supply a MID name to be used for formatting the next input message. It is the responsibility of the workstation to supply this MID name when it sends the input message. This results in supplying the MID name.

MID/MOD chaining can be accomplished with little or no intervention by the workstation operator by observing the following procedure during the design of SLU P system application programs:

1. Remove the MID name from the received output message header and save it for use on the next input message.
2. Display or print the output message.
3. Get the next operator input.
4. Add the MID name saved in step 1 to the transaction.
5. Send the transaction to IMS.

Related tasks

“Activating MFS input formatting for Finance workstations” on page 936
When MFS is used, input messages can be processed by the message and format descriptors.

“MID/MOD chaining” on page 912
MFS input formatting for a workstation occurs when a message input descriptor (MID) name is provided with an input message.

MFS output formatting for the SLU P system

MFS can be used to format pages, produce standard headings and footings, and provide forms control. Without MFS, the message processing program (MPP) or the controller application program must provide these functions.

Using MFS can also provide MPP device independence and allow SLU P components to be used for low-volume applications invoked from any of several terminal types. This device independence allows the controller application program to concentrate on high-volume applications, reducing its complexity and maintenance work.

The availability of MFS to a workstation is defined on a station-by-station basis by the system administrator during IMS system definition or ETO logon descriptor definition.

MFS message recovery

Input messages processed by MFS are edited before the message is queued and logged. Output messages are edited immediately prior to transmission; therefore, for recovery purposes, an input message is formatted by MFS and an output message is formatted by the IMS application program.

MFS control functions (Finance)

The operators of devices using MFS can use a number of control functions.

The control functions available to the operators of devices using MFS are:

Page Advance (NEXTPP)
Transmit the next physical page of the current message, if one exists.

Logical Page Advance (NEXTLP)
Transmit the first or only physical page of the next logical page of the current message.

Logical Page Requests
PAGEREQ=nn where nn is the number of the logical page desired. If the request is valid, transmit the data fields defined in the specified logical page of the current message.

Message Advance Protect (NEXTMSGP)
Discontinue printing or displaying the current output message and begin transmitting the next message in the output queue. If none exists, return an exception DR2 to notify the operator.

Message Advance (NEXTMSG)
Discontinue printing or displaying the current output message and begin transmitting the next message, if any.

These control functions can be indicated to MFS by including them in input function management headers or by defining an operator control field within the input data to MFS.

MFS control functions (SLU P)

The following device-level MFS control functions are available to logical units defined as SLU P.

SLU P terminals process the DPM-formatted output messages that specify paging option on the MFS device format. The paging option can be OPTIONS=DPAGE or OPTIONS=PPAGE.

Page Advance (NEXTPP)
If a current message with OPTIONS=PPAGE is defined, transmit the data fields defined in the next presentation page. If a current message with OPTIONS=DPAGE is defined, transmit the data fields defined in the next logical page.

Logical Page Advance (NEXTLP)
Transmit the data fields defined in the next logical page of the current message if one exists and if either OPTIONS=PPAGE or OPTIONS=DPAGE is specified.

Logical Page Request
PAGEREQ=nn, where nn is the number of the logical page desired. If the request is valid, transmit the data fields defined in the specified logical page of the current message.

Message Advance Protect and Message Advance (NEXTMSGP and NEXTMSG)
Discontinue transmitting the current output message and begin transmitting the next message in the output queue.

The control functions of NEXTPP, NEXTLP, NEXTMSGP, and NEXTMSG can be specified in input function management headers or by defining an operator control field within the input data to MFS. Logical page requests can only be entered within the input data or using the operator control field.

MFS paging and BID options

The BID option provides output protection but has a high performance cost (an extra line flow for each message).

When the BID option is specified and the output occurs while between-brackets, IMS waits for a positive response to BID before sending output.

To avoid sending BID while sending MFS-paged output, each input paging request should indicate begin-bracket and change-direction. Each output page then contains only end-bracket and no BID results, because the page is sent while in an in-brackets state.

Display screen protection for finance stations

When a Finance station is defined with the NOBID option, IMS provides support similar to the screen protection function for the IBM 3270 Information Display System.

IMS does not transmit two consecutive output messages to a display component without an intervening input message from the component. This gives the user the opportunity to view and respond to one message before another is displayed.

When IMS transmits an output message to a display component, it marks the component as protected, unavailable for output. IMS does not transmit another output message to it until an input message is received from that component. The input message can be an IMS transaction, a message switch, an IMS command, the VTAM RTR command, or an MFS control request. Upon receipt of one of these, IMS changes the display component's status to unprotected.

When a workstation is defined with the BID option, consecutive messages are not transmitted to a station. Input from the workstation or a positive response to the BID command from IMS must be received after one transmission before the next message can be sent.

When a workstation uses MFS, the screen is protected on a physical-page basis. MFS control requests or input data are used to request additional screens of data.

If a workstation does not use MFS, the screen is protected on a message-by-message basis. The input of any IMS transaction, message switch, or command causes the screen to be unprotected. The VTAM ready-to-receive (RTR) command can be used to request the next output message. If no message is available, or if the node is in a status that does not allow output to be sent (output stopped or quiesced), IMS returns an exception DR1, followed by an error message indicating that no output is available.

If the RTR command is received during the transmission of MFS-paged output, IMS returns an exception DR1 response, followed by an error message indicating an invalid paging request, because IMS cannot determine the MFS control function to be performed.

After an exception response is received by IMS for a current IMS output message to a display component, the screen is unprotected. After an exception response is sent by IMS, any defined display component is automatically marked protected and unavailable for output.

Related reference

“MFS control functions (Finance)” on page 913
The operators of devices using MFS can use a number of control functions.

Extended output component protection (SLU P)

When a SLU P is defined with the NOBID option, IMS provides support similar to the screen-protection function for the IBM 3270 Information Display System and display-screen protection for Finance components.

IMS does not transmit two consecutive output messages to an output-protected component without an appropriate intervening input message from the LU. This gives the LU an opportunity to process one message, according to user-defined procedures, before another is sent to the same component. SLU P support allows the output of all components to be protected.

IMS marks the component as protected and unavailable for output under the following conditions:

- When IMS transmits a message to a component defined as PROGRAM2 on the IMS TERMINAL macro
- When IMS transmits a message (or a page of a message) formatted by MFS with a paging option defined

While a component is protected, IMS does not transmit another output message to the component until an input message from the logical unit resets the component to unprotected. Input messages that can reset a component's status to unprotected are:

- An IMS transaction with an FM header
- A message switch with an FM header
- An IMS command with an FM header
- An MFS control request with an FM header indicating a specific component to be unprotected
- Any of these, either without an FM header or with an FM header indicating component zero (which resets protection on all components of a terminal)
- An RTR command

After the component's status is reset to unprotected, IMS sends any available output. If a component is defined as PROGRAM2 and is not described by a device format with paging OPTIONS=DPAGE or OPTIONS=PPAGE, output to the component is protected on a message-by-message basis. The input messages are used to reset output component protection.

If a SLU P component is defined to use MFS DPM and the device format used is defined with the paging option OPTIONS=DPAGE or OPTIONS=PPAGE, the output component is set to protected when a logical page or presentation page is sent to the component. MFS control requests or input data are used to reset output component protection and request additional logical or presentation pages from MFS.

During the transmission of MFS-paged output, the only allowable input is:

- Input containing no message header (resets all components' protection).
- Input containing a header indicating the specific component to which MFS is currently paging. Otherwise, the session is terminated, because IMS cannot simultaneously send output to more than one component of a terminal.

If a SLU P terminal is defined with the BID option, output component protection occurs after every message. In this mode of operation, consecutive messages are not transmitted to a terminal. Input from the terminal must be received; otherwise, if additional output is queued, a BID command is sent after one transmission before the next message is sent.

If a terminal does not use MFS, screen protection is performed on a message-by-message basis. The input of any IMS transaction, message switch, or command causes the screen to be unprotected.

If the ready-to-receive (RTR) command is received and no messages are available, or if the terminal is in a status that does not allow output to be sent (output stopped or quiesced), IMS returns an exception DR1, followed by an error message indicating no output is available. If the RTR command is received during the transmission of MFS-paged output, IMS returns an exception DR1 response, followed by an error message, indicating invalid paging request because IMS cannot determine the MFS control function to be performed.

After any exception response is sent by IMS, any display component defined for the logical unit is automatically marked protected and unavailable for output.

After an exception response is received for a current IMS output message to a component defined as PROGRAM2 or for an MFS DPM-paged output message, the component to which the output was sent is left unprotected.

After any exception response is sent by IMS, all components defined as PROGRAM2 are automatically marked protected and unavailable for output.

Related reference

“MFS control functions (SLU P)” on page 913

The following device-level MFS control functions are available to logical units defined as SLU P.

Input and output editing options (SLU P)

The type of editing subparameter in the COMPTn parameter of the system definition TERMINAL macro and the Extended Terminal Option (ETO) logon descriptor can be indicated at a component-by-component level for input to and output from components defined as SLU P.

The four types of editing provided are:

BASIC
Requests that no deblocking be performed on input to IMS. MFS cannot be used on input or output.

BASIC-SCS1
Requests that deblocking occur when an SNA character string (SCS)-defined new line (NL-X'15') or form feed (FF-X'0C') character is sent to IMS. MFS is not used on input or output.

MFS-SCS1
Requests that deblocking occur when an SCS-defined new line (NL-X'15') or form feed (FF-X'0C') control character is sent to IMS. MFS-SCS1 formats can be used for both input and output.

DPM-An
Allows messages to be formatted using MFS distributed presentation management (DPM). No deblocking on input to IMS is performed. ‘n' is a number from 1 to 15.

For DPM, the device type symbolic name is specified to MFS as DPM-Xn, where X is A or B. Message formatting is specified on the TERMINAL macro for logical units defined as SLU P by using device type symbolic names of the form DPM-An. DPM-Bn device type symbolic names refer to logical units defined for Intersystem Communication (ISC).

If BASIC or SCS1 is specified, each message received or sent by IMS as one or more related transmissions forms a VTAM chain:

- For BASIC, each input transmission to IMS, less any supplied input FM header, is treated as an IMS segment to be presented to MFS or directly to the IMS message processing program.
- For SCS1, an IMS input segment is created at each SCS-defined new line or form feed control character. IMS segments can be created from a portion of, from all of, or from more than one spanning transmission.

For either BASIC or SCS1, each output segment is sent by IMS in a single transmission, unless it and any IMS-appended FM header are larger than the logical unit's buffer size as indicated to IMS by the OUTBUF parameter on the system definition TERMINAL macro or on the ETO logon descriptor. In this case, IMS sends the segment in as many transmissions as required. Each transmission, except possibly the last transmission of a segment, is the maximum-defined output buffer size.

When DPM or MFS-SCS1 is specified, input and output messages can be formatted by MFS DPM or MFS-SCS1, respectively, on a message-by-message basis. Messages that are not to be formatted are edited as previously described for BASIC or SCS1.

Related concepts
MFS message formatting functions (Application Programming APIs)
IMS Message Format Service (Application Programming)
Related tasks
“Administering the Extended Terminal Option” on page 69

The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

Use of responses or brackets to acknowledge recoverable input

To facilitate message resynchronization, IMS allows an input update or a recoverable-inquiry transaction to optionally request a definite response.

IMS allows this request for a definite response if the user specifies the OPTIONS=OPTACK for the workstation.

With this operand specified, IMS can acknowledge input with the next output through use of the input and output bracket indicators.

Restriction: OPTIONS=ACK is not supported for SLU P terminals created using the ETO feature.

If the OPTACK option is defined, performance can be improved if the workstation requests begin-bracket, change-direction, and exception DR1 or DR2 (rather than DR1 or DR2) on recoverable input to IMS. The next output from IMS acknowledges the input by indicating end-bracket only. The type of output messages that are sent from IMS depend on the type of recoverable input, the defined response mode, and the availability of output.

Contention can occur when a session is in a between-brackets state. In this case, IMS sends an unsolicited output message indicating both BB and EB or a BID command at the same time that the workstation is sending an input message indicating begin-bracket and change-direction. This output message does not acknowledge the input message and can be either accepted or rejected by the workstation. Rejecting an IMS nonrecoverable output message can result in losing the message. If contention occurs when IMS is sending a recoverable IMS output message (output requesting DR2 response), the workstation must send either an exception or a definite response before IMS can receive the input message.

Related concepts

“Message resynchronization” on page 918
The purpose of message resynchronization is to guarantee the integrity of messages across sessions.

“Output messages sent while in a between-brackets state” on page 910
If an IMS output message is sent while the workstation is not protected from output and in a between-brackets state, IMS sends one of two things.

Related tasks

“Administering the Extended Terminal Option” on page 69
The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

Related reference

“Fast Path messages with Finance and SLU P” on page 919
When using Fast Path, you must specify certain options on the TERMINAL system definition macro.

“SLU P message protocols” on page 933
A single transmission must be used for VTAM commands and indicators and MFS control requests. Single or multiple transmissions can be used to send IMS transactions, commands, and message switches.

“MFS control functions (Finance)” on page 913
The operators of devices using MFS can use a number of control functions.

“TERMINAL macro” on page 483

Several system definition keyword parameters on the TERMINAL macro are principal for defining an ISC session.

Message recovery

The process of message recovery is accomplished within IMS by using the IMS system log and the checkpoint and restart routines.

Recovery is possible, because IMS has direct control of the communication link. With a SLU P system, however, this control is shared with the controller and the controller application program. If a network failure, such as a processor failure or an IMS abend, occurs while the controller application program is receiving a message from IMS, the traditional methods of IMS recovery cannot always detect the lost message. The IMS recovery methods are extended for the SLU P systems to include the controller application program for message resynchronization.

The controller application program participates in message resynchronization so that a lost message condition, if any, can be detected and corrected. Failure of the controller application program to perform its responsibilities during message resynchronization results in a loss of message integrity in the system.

Message resynchronization is necessary for a workstation when a session with the workstation is terminated between the time it sends a recoverable message and the time it receives a reply for that message. The type of resynchronization depends on how the message is defined.

For all messages except response mode, response conversational mode, and Fast Path, the input becomes recoverable when it has been successfully enqueued and made available for scheduling. During message resynchronization, IMS indicates the last successfully received recoverable message.

For response mode, response conversational mode, and Fast Path, the input is not made recoverable and restartable until the application reaches its first sync point. If the IMS system fails before this sync point, the message is considered to be nonrestartable. During message resynchronization, IMS indicates whether the message has reached a sync point or needs to be reissued.

If a session with a workstation is terminated after IMS sends a message to the workstation but before IMS receives the response, message resynchronization is necessary for this workstation. The output message for which no response is received must remain associated with this workstation until message resynchronization determines whether the workstation received the message. If the **/ASSIGN** command is used to move the message to a different workstation, message resynchronization is no longer possible.

Message resynchronization

The purpose of message resynchronization is to guarantee the integrity of messages across sessions.

Message resynchronization occurs at the start of a session unless IMS is cold started. Finance and SLU P sessions warm start using the control blocks created from the original descriptor, even though that descriptor might have been changed or deleted after IMS created the control blocks. Only a cold start ensures that the control blocks that are created represent the new or updated descriptor.

When message resynchronization is necessary because of a network failure, the resynchronization must complete successfully before IMS permits normal data transmission. To initiate message resynchronization, IMS sends the VTAM set-and-test-sequence-numbers (STSN) command. The LU must respond to this command. To be able to respond properly, a copy of the following must be maintained:

- The sequence number of the last request unit (RU) of the last inbound sync-point message that was sent by the LU. The inbound sync-point message is one of the following:
  - The last successful recoverable input message (the one that requested a DR1 or DR2) if the ACK option was defined to IMS for this LU
  - The last input message, if the OPTACK option was defined for this LU
- The sequence number of the last request unit (RU) of the last outbound sync-point message received by the LU. The outbound sync-point message is the last successful recoverable output message. Recoverable output messages are those requesting DR2 responses.

If Fast Path is used, each output message is recoverable and requests an exception DR2. These Fast Path-recoverable messages are identified by a flag within the output message header, rather than by an explicitly requested DR2 response.

Optionally, the LU can maintain a copy of the last inbound recoverable message sent by the terminal. If this is done, the SLU P system can retransmit, after resynchronization, any message not received by IMS.

Restriction: Session initiation and resynchronization caused by an SNA Request-Recovery (RQR) command is not allowed when the node is in response mode and the response reply message is not yet available for output; that is, the input response mode transaction is still queued or in the process of execution. Response-mode transactions are not recoverable or restartable prior to the application sync point; therefore, session input acknowledgment does not occur until input processing is complete.

Session initiation or resynchronization results in session termination while the response mode transaction is in this 'indoubt' state, and IMS is not able to indicate that the associated input sync-point request was committed or backed out. This temporary error condition is indicated to the master terminal operator by message DFS2081I. An IMS **/DISPLAY** command can be used to determine when the response mode reply message is available and session initiation can again be attempted. A display status of RESP-INP indicates input is still in process; RESP indicates input processing is complete and output is available for transmission.

Finance and SLU P in an XRF complex

Finance and SLU P sessions in an XRF complex are handled as Class 1, 2, or 3 terminals.

Class 1 attempts to provide a maximum level of transparency and performance by tracking the active IMS session with a standby session on the XRF backup system. Class 2 and 3 support involves terminating the active session if the IMS system fails. At takeover, the alternate IMS system automatically reinitiates the session and automatically signs on the original active system user if necessary. Class 3 support requires manual session restart and signon after XRF takeover.

For information about XRF from a z/OS perspective, see *z/OS Communications Server: SNA Network Implementation Guide*.

Related concepts

Extended Recovery Facility Overview (System Administration)

Related tasks

"Establishing connection with the XRF complex" on page 905

To establish a session with IMS in an XRF complex, your logon request must include either the USERVAR name you defined in the USERVAR tables or the MNPS ACB name shared by both of the IMS systems in the XRF complex.

Fast Path messages with Finance and SLU P

When using Fast Path, you must specify certain options on the TERMINAL system definition macro.

The following options must be specified in the TERMINAL macro:

OPTACK
Allows input messages containing begin-bracket and change-direction indicators to be acknowledged by an end-bracket indicator on the next output message.

FORCERESP or TRANSRESP
Allows input of a response mode message to IMS.

To obtain best performance, Fast Path input messages should not request definite responses but should be coded with begin-bracket, change-direction, and exception DR2 response.

FPACK/NFPACK
Determines if special Fast Path output protocols should be used.

Fast Path transactions are single-segment transactions and are defined as recoverable. Fast Path transactions must be defined as response mode.

Fast Path output messages (Finance)

When defined with the system definition TERMINAL macro option FPACK or an ETO logon descriptor, Fast Path output messages are sent requesting an exception DR2 response.

Fast Path output is sent requesting DR2 response when:

- Queued output is available to be sent to any nondisplay component.
- The system definition TERMINAL macro option BID or an ETO logon descriptor option BID is defined, and queued output is available to be sent to a display component.
- The system definition TERMINAL macro option NOBID or an ETO logon descriptor option NOBID is defined, and queued output is available to be sent to a display component when the Fast Path output reply message is directed to a nondisplay component. The Fast Path output does not result in the display component being screen protected.

Unlike non-Fast Path output, the output message is left outstanding (not dequeued), and the workstation remains in response mode until the workstation sends data, a ready-to-receive (RTR) command, or the DR2 response to cause the message to be dequeued. The RTR command should be considered when no input is to be generated for an abnormally long time (for example, when the terminal operator plans to leave the terminal). The input message from the workstation, the RTR command, or the DR2 response acknowledges that the preceding output has been received and is recoverable; therefore, IMS might dequeue the output message, process any input message, or send any available output.

The system definition TERMINAL macro option NFPACK or an ETO logon descriptor indicates that the Fast Path output exception DR2 and next input acknowledgment protocol should not be used. In this case, Fast Path output messages are always sent requesting standard recoverable output message response protocols (DR2 response).

When the DR2 response is received, IMS dequeues the output message, removes the terminal from response mode, and sends any available queued output. Queued output is not sent to a display component if the Fast Path output reply message is sent to the same component because of the IMS screen protection support.

Related concepts

“Display screen protection for finance stations” on page 914
When a Finance station is defined with the NOBID option, IMS provides support similar to the screen protection function for the IBM 3270 Information Display System.

Fast Path output messages (SLU P)

When defined with the system definition TERMINAL macro option statement FPACK or an ETO logon descriptor, Fast Path output messages are sent requesting an exception DR2 response when certain events occur.

Fast Path output messages request an exception DR2 response when any of the following occurs:

- Queued output messages are available to be sent to a component defined for the workstation as PROGRAM1.
- The system definition TERMINAL macro option BID or an ETO logon descriptor option BID is defined, and queued output is available to be sent to a component defined for the workstation as PROGRAM2.
- The system definition TERMINAL macro option NOBID or an ETO logon descriptor option NOBID is defined, and queued output messages are available to be sent to a program. This component is not output protected due to a previous output or this Fast Path message. (That is, multiple components might be protected or reset selectively as defined in this chapter.)

Unlike non-Fast Path output, the output message remains outstanding (not dequeued) and the terminal remains in response mode until the terminal sends data, a ready-to-receive (RTR) command, or the DR2 response to cause the message to be dequeued. The RTR command should be considered when no input is to be generated for an abnormally long time (for example, when the terminal operator plans to leave the terminal). The input message from the terminal, the RTR command, or the DR2 response acknowledges

that the preceding output has been received and is recoverable; therefore, IMS might dequeue the output message, process any input message, or send any available output.

The system definition TERMINAL macro option NFPACK or an ETO logon descriptor indicates that the Fast Path output exception DR2 and next input acknowledgment protocol should not be used. In this case, Fast Path output messages are always sent requesting standard recoverable output message response protocols (DR2 response).

When the DR2 response is received, IMS dequeues the output message, removes the terminal from response mode, and sends any available queued output for PROGRAM1 components or non-output protected PROGRAM2 components defined for the workstation. Output protection is set at the component level during output for PROGRAM2 components and can be selectively reset for one or more components based on subsequent input.

Also use the NFPACK option for all 4730 terminals. This allows any asynchronous output messages to be sent immediately following the acknowledgment of the Fast Path output reply.

Related concepts

“Extended output component protection (SLU P)” on page 914
When a SLU P is defined with the NOBID option, IMS provides support similar to the screen-protection function for the IBM 3270 Information Display System and display-screen protection for Finance components.

Fast Path message resynchronization

If Fast Path is defined, sequence number management and message resynchronization might require change.

Related concepts

“Message resynchronization” on page 918
The purpose of message resynchronization is to guarantee the integrity of messages across sessions.

Chapter 51. Network operation for SLU P and Finance

The following topics describe how to start an IMS network, how to initiate sessions, the different transaction types, message switching, and IMS commands.

Starting an IMS network

Before a session with IMS can be established for any workstation VTAM, NCP, controllers, and logical units must be active.

Specifically, the following components must be active:

- VTAM and NCP must be active.
- Controllers and logical units that are not activated automatically by VTAM must be activated with a **VARY** command by the VTAM network operator.
- Controllers must be powered on, and appropriately configured, initialized, and activated for VTAM and NCP.

Making IMS ready

IMS must be made ready to receive VTAM logon (session initialization) requests. Ready IMS by issuing the IMS **/START** command with the DC keyword.

The **/START DC** command tells VTAM to pass any queued VTAM logon requests (and any logon requests for workstations known to VTAM as belonging to IMS) to IMS.

The **/START DC** command activates the following processes:

- Initiates IMS Transaction Manager processing
- Opens the VTAM access method control block
- Enables the IMS VTAM logon exit

Any logon requests received by VTAM before the IMS **/START DC** command is issued, but after the IMS VTAM access method control block (ACB) has been opened, are queued in VTAM until the **/START DC** command processing is completed. If VTAM is active when IMS is initialized, and the DFSDCxxx PROCLIB member keyword VACBOPN=INIT, then the IMS VTAM ACB is opened. If DFSDCxxx PROCLIB member keyword VACBOPN=DELAY, then the IMS VTAM ACB open is delayed until the **/START DC** command is processed.

Session initiation (starting workstations)

A *session* is the logical connection of a workstation to a VTAM application program, such as IMS or the system utilities. A session must be established before data can be transmitted between a workstation and IMS. Message resynchronization is performed during session initiation, unless IMS is cold started.

Session initiation can be requested in one of the following ways:

- The workstation requests session initiation by sending the **INITIATE-SELF** command. VTAM verifies the command and passes the request to IMS. If the terminal is requesting a session with an IMS XRF complex, the IMS USERVAR name or the MNPS ACB should be used for the application name.
- The z/OS VTAM network operator requests session initiation on behalf of the workstation by using the VTAM **VARY** command with the LOGON option. VTAM processes the request and passes it to IMS.
- VTAM passes a logon request to IMS for each workstation that is defined to VTAM as belonging to IMS.
- The IMS master terminal operator requests session initiation for a workstation by entering the IMS **/OPNDST** command.

© Copyright IBM Corp. 1974, 2020

Regardless of how session initiation is requested, identical processing occurs when IMS receives the request.

Session-initiation transmission sequence

The following list shows the sequence of transmissions that occurs when a workstation requests session initiation.

The numbers relate to major events in the sequence.

1. The controller sends the **INITIATE-SELF** command. VTAM checks the validity of the command, looking for syntax errors that might have been introduced by either the controller or NCP.
2. If the command is not valid or if VTAM does not find a match when verifying the initiation request, VTAM returns an EXC/DR1 response. EXC/DR1 terminates the initiation request. The SLU P system can retry the initiation request or notify the operator to begin corrective action.

   If the command is valid, VTAM returns a DR1 response. VTAM then verifies the content of the resource field in the **INITIATE-SELF** command. This field must contain the name of the VTAM application program with which the workstation wants to enter a session. VTAM compares the resource field content to its list of active application programs (those programs with an open VTAM ACB).
3. If a match is found, VTAM passes the logon request to the specified application program (IMS).

   In an XRF environment, VTAM routes the request to the currently active system.

   When IMS receives the request, it compares the workstation node name supplied by VTAM to the node names defined during IMS system definition.
4. If IMS does not recognize the workstation, it issues a VTAM **CLSDST** command, which causes VTAM to send a procedure error command. A procedure error terminates the session initiation request. The system notifies the workstation operator that the session is denied.

   If IMS recognizes the workstation, it issues a VTAM **OPNDST** command, which causes VTAM to send a **BIND** command. The **BIND** command contains the name of the application program (IMS) that issued the **OPNDST** command.

   In an XRF complex that uses USERVAR instead of MNPS, the **BIND** command from VTAM contains the USERVAR segment in the user data field and the active APPLID in the PLUNAME field.

   IMS supplies a set of **BIND** parameters, which define the communication rules and protocol that must be followed. IMS ignores the parameters of the mode table entry supplied by the workstation or network operator.

   The workstation must respond to the **BIND** command.

   Related reading: For more information on the contents of the **BIND** data, see "Bind Parameters for SLU P and LU 6.1" in *IMS Version 15 System Programming APIs*.
5. If the workstation cannot begin a session, it returns an EXC/DR1. This can occur when either VTAM or IMS is requesting session initiation and the workstation is currently unable to communicate (for example, because it is involved in offline processing).

   If the workstation can begin a session, it returns a DR1. When IMS receives the DR1, it performs message resynchronization, if necessary.
6. If message resynchronization is not necessary, or when it is complete, IMS sends a VTAM start-data-traffic (**SDT**) command.

   The workstation is in session and can transmit data to IMS.

Related reference

Bind parameters for SLU P and LU 6.1 (System Programming APIs)

Controller application program involvement in message resynchronization

The controller application program must participate in message resynchronization and, in order to do so, must have maintained copies of the sequence numbers of the last recoverable message that

was successfully sent to IMS and the last recoverable message that was successfully received by the workstation.

Design considerations

The system application analyst must determine where to maintain the sequence numbers necessary for set-and-test-sequence-numbers (STSN) processing.

Three options are available:

- The controller disk or diskette.

  The disk or diskette in the controller is the most reliable method and does not require involvement of the workstation operator if retransmission of a message is required. Either the permanent file or the transient files of the disk or diskette can be used. The REPLACE instruction should be used when writing to the disk or diskette file. REPLACE causes the data to be physically written to the disk or diskette and not just buffered in the controller's control storage. Disk or diskette storage assures sequence number retention across a power failure or a controller failure that can destroy the contents of controller's control storage. However, disk or diskette access and data transfer time can add significant overhead to each recoverable transaction.
- The controller's control storage.

  If the controller's control storage is used to retain sequence numbers, a power loss or controller failure would destroy the contents of the controller's storage, making message resynchronization impossible. By not performing message resynchronization, recoverable input and output messages can be lost or duplicated.
- A workstation output device.

  Message sequence numbers can be stored on a workstation output component such as the display or the journal printer. However, this method requires workstation-operator intervention to retrieve the numbers during STSN processing.

Sequence number management

An understanding of how sequence numbers are handled during normal message transmission is prerequisite to understanding how and why the SLU P system performs the functions described in this topic.

The management of sequence numbers is shared between the controller, VTAM, and the VTAM application program (IMS). The controller assigns sequence numbers to messages originated by its workstations. VTAM assigns VTAM sequence numbers to messages originated by IMS. To ensure recoverability, IMS maintains a separate copy of the sequence numbers associated with the messages it sends.

Sequence numbers are assigned to each workstation. For each workstation, the controller tracks the last sequence number that it assigns (called the *last-assigned value*), and tracks the last sequence number that it receives (called the *last-received value*). Similarly, IMS tracks the last sequence number that VTAM assigns (called the *last-assigned value*), and tracks the last sequence number it receives (called the *last-received value*).

When the controller issues a request to transmit data, it updates its last assigned value for the requesting workstation, appends the new number to the data, and sends the message. When VTAM receives the message, it merely passes it on to IMS. IMS, in turn, adds 1 to its last received value for that workstation and compares this value with the sequence number of the message it just received. If the two numbers match, IMS processes the message as required. If the two numbers do not match, IMS issues a VTAM CLSDST macro instruction to terminate the session.

When IMS has a message to transmit, it updates its copy of VTAM's last assigned value and sends the message to VTAM. VTAM updates its last assigned value, appends the new number to the message, and sends the message. The controller removes the appended sequence number, updates its last received value, and compares this value with the sequence number that accompanied the message. If the two numbers match, the controller sends the message on to the application program. If the two numbers do

not match, the controller returns an exception DR2 to indicate a sequence error. When IMS receives the sequence error indication, it issues a VTAM CLSDST macro instruction to terminate the session.

Set-and-Test-Sequence-Numbers (STSN)

Message resynchronization is initiated by IMS when it sends the set-and-test-sequence-numbers (STSN) command to the workstation.

The STSN command contains a 5-byte data field:

Byte 0
Action code

Bytes 1, 2
Controller sequence number of the last inbound sync-point message IMS received from the workstation

Bytes 3, 4
VTAM sequence number of the last outbound sync-point message IMS sent to the workstation

IMS uses the action code to ask the SLU P system to verify the controller and VTAM sequence numbers. The bits of the action code byte are:

Bits 0, 1
Refer to the controller sequence-number field

Bits 2, 3
Refer to the VTAM sequence-number field

Bits 4, 5, 6, and 7
Reserved

The following values are acceptable for bits 0, 1, 2, and 3 of the action code:

00 IGNORE
Not used by IMS.

01 SET
Set the appropriate sequence number to the value indicated in the sequence-number field.

10 INVALID
Not used by IMS. The SLU P system must return its version of the sequence number in the command response.

11 SET AND TEST
Set the appropriate sequence number to the value indicated in the sequence-number field. The SLU P system must indicate in the command response whether the sequence number values are acceptable.

IMS uses the SET option for the controller sequence number. For the VTAM sequence number, IMS uses either SET or SET AND TEST. SET is used when no acknowledgment to a recoverable output message is outstanding from the previous session. SET is also used when the IMS master terminal operator enters the **/DEQUEUE** or **/ASSIGN** commands that cause the last recoverable message to be removed from the queue. SET AND TEST is used if IMS sends a recoverable message to the workstation but does not receive the required acknowledgment prior to session termination. The response sent by the SLU P system to the STSN command indicates whether the workstation received the message.

When the SLU P system receives the STSN command, it must be able to:

- Verify the controller sequence number and arrange to retransmit a message to IMS if required.
- Verify the VTAM sequence number and inform IMS whether the number is acceptable.
- Return a DR1 to IMS and, if required, return a 5-byte data response to the STSN command.

To verify the controller sequence number, the SLU P system compares the number provided by IMS with the number it maintained from the previous session. An equal compare indicates that IMS received all messages sent by the workstation. If the IMS-provided number is smaller, IMS did not receive the last inbound sync-point message sent by the workstation. These two numbers can differ by more than 1 if intervening irrecoverable or chained messages were sent by the workstation when the workstation was

defined with the ACK option. A message that was not received by IMS should be retransmitted after message resynchronization is complete. If a copy of the message was maintained, that copy could be sent. Otherwise, the SLU P system can inform the workstation operator that the last inbound recoverable message was not received by IMS.

To verify the VTAM sequence number, the SLU P system compares the number provided by IMS with the number it maintained from the previous session. An equal compare indicates that the workstation has received all messages. If the IMS-provided number is unequal, the workstation did not receive the last outbound recoverable message.

The response to the STSN command indicates the results of the synchronization tests to IMS. The SLU P system must return a DR1 and, optionally, 5 bytes of data. If both the controller and VTAM sequence numbers are acceptable, only the DR1 is required. If one or both of the sequence numbers are not acceptable, a DR1 and the 5-byte data response are required. The format of the data response has the same format as the STSN command:

Byte 0
Action code

Bytes 1, 2
Controller sequence number of the last inbound recoverable message the workstation sent to IMS

Bytes 2, 3
VTAM sequence number of the last outbound recoverable message the workstation received from IMS

The SLU P system uses the action code to indicate the test results. The bits of the action code byte are:

Bits 0, 1
Refer to the controller sequence-number field

Bits 2, 3
Refer to the VTAM sequence-number field

Bits 4, 5, 6, and 7
Reserved

The following values are acceptable for bits 0, 1, 2, and 3 of the action code:

00 RESET
The sequence number in the command is unacceptable and should be reestablished at its previous value. This code should never be returned to IMS.

01 TEST POSITIVE
The sequence number in the command is acceptable. This code must be returned in response to the SET option. If the SET AND TEST option is specified and the SLU P system finds the sequence number acceptable, TEST POSITIVE should be returned.

10 INVALID
The SLU P system has detected an error in the sequence number.

11 TEST NEGATIVE
The SLU P system does not agree with the sequence number presented with the SET AND TEST option.

TEST POSITIVE is the only acceptable response to the SET option. TEST POSITIVE and TEST NEGATIVE are acceptable responses to the SET AND TEST option. INVALID should be used if the SLU P system detects a "should-not-occur" condition when comparing the sequence numbers. An example of this condition is a controller sequence number specified in the command that is higher than the application program's copy of the sequence number.

If IMS receives a TEST NEGATIVE response to the SET option, a RESET response, or an INVALID response, it terminates the session.

When returning any action code except TEST POSITIVE, the SLU P system should use the STSN response to return its version of the sequence numbers. The sequence numbers can be valuable information during problem determination and debugging. Any value can be returned, because IMS does not use the

returned values. A TEST NEGATIVE response is all that is required to indicate that the sequence number is not acceptable.

When IMS receives the STSN response, it sends the start-data-traffic (SDT) command to the workstation. The SDT command allows normal message transmission to begin. If IMS receives a TEST NEGATIVE response to the SET AND TEST option, it sends the SDT command and retransmits the last recoverable message followed by any other available output. When IMS receives a TEST POSITIVE response to the SET AND TEST option, it sends the SDT command, dequeues the last recoverable message, and sends any other available output.

The contents of the STSN command as received from IMS and the resulting actions of the SLU P system and VTAM are summarized in the following table.

Table 176. Set-and-Test-Sequence-Numbers (STSN) summary

Remote system number action code	VTAM number action code	Controller sequence number^{"1" on page 928}	VTAM sequence number^{"1" on page 928}	Remote system action when STSN received	VTAM action when STSN received
01 set		Last inbound sync point message that IMS received		Set controller sequence number field to value indicated in STSN controller field.	Set VTAM value to value indicated in STSN controller field.
	01 set		Last outbound sync point message that IMS sends, and responded to by station.	Set host sequence number field to value indicated in VTAM field of STSN.	Set VTAM's last assigned value to value in VTAM field of STSN.
	11 set and test		Last recoverable message that IMS sends; no response from station received.	Set host sequence number field to value indicated in VTAM field of STSN.	Set VTAM's last assigned value to value in VTAM field of STSN.

Note:

1. The controller and VTAM sequence number fields can contain any value.

The valid SLU P system responses to the STSN command and the resulting IMS actions are summarized in the following tables.

Table 177. STSN response summary when the action code is the remote system number

Remote system action code value	Response action code must be^{"1" on page 928}:	IMS action when response received
01 set	01 test positive	Send SDT to station

Note:

1. Any other response causes IMS to terminate the session.

Table 178. STSN response summary when the action code is the VTAM number

VTAM number action code value	Response action code must be^{"1" on page 929}:	IMS action when response received
01 set	01 test positive	Send SDT to station

Table 178. STSN response summary when the action code is the VTAM number (continued)

VTAM number action code value	Response action code must be"1" on page 929:	IMS action when response received
11 set and test	01 test positive	Dequeue last recoverable message and send SDT to station
11 set and test	11 test negative	Send SDT to station and retransmit last recoverable message.

Note:

1. Any other response causes IMS to terminate the session.

Suspending output from IMS

If the controller application program does not want or cannot receive any more output from IMS, the program can send the VTAM quiesce-at-end-of-chain (QEC) command to IMS. IMS returns a DR1 and the VTAM quiesce-complete (QC) command. IMS does not send any more output to the workstation until it receives the VTAM release-quiesce (RELQ) command.

A workstation can also suspend output from IMS by sending the VTAM **LUSTATUS** or **SIGNAL** commands to IMS.

Session termination

Session termination releases the workstation from its current logical connection to the VTAM application program. It makes the workstation available for session with other VTAM applications, or communications can be terminated altogether.

There are two types of session termination: orderly and immediate.

Definitions:

- In an *orderly termination*, the workstation is allowed to complete normal processing before the session is terminated.
- An *immediate termination* forces the workstation to terminate the session unconditionally.

Session termination can be invoked by any of the following:

The IMS master terminal operator
The VTAM network operator
The workstation

The following figure summarizes the two types of session termination processing.

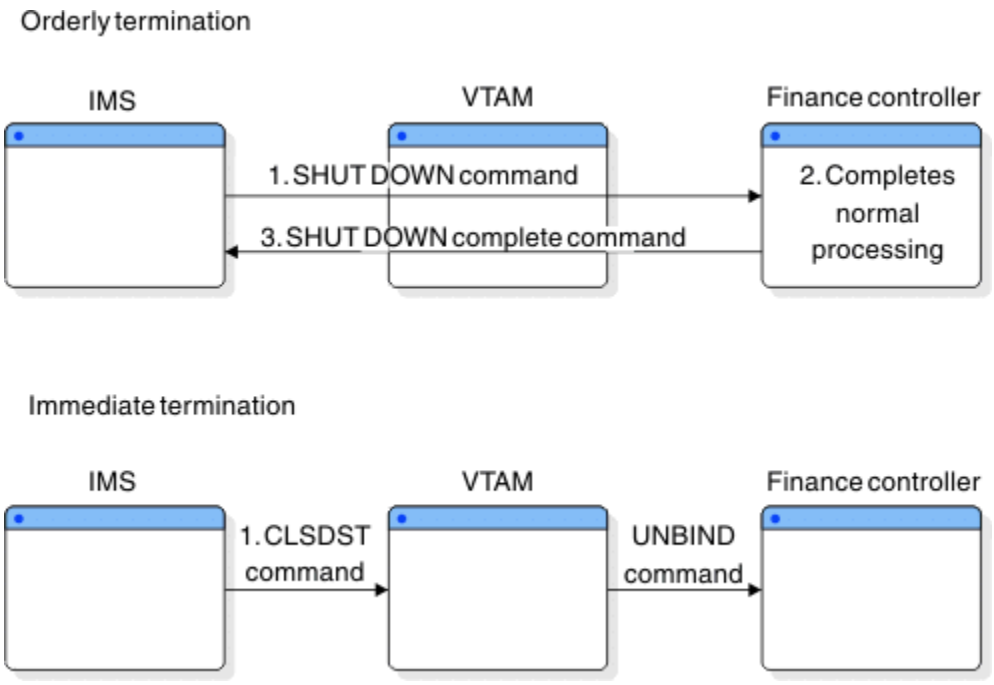


Figure 158. Termination processing

Each installation must determine specific procedures for session termination. When developing the procedures, be aware of the requirements for session termination processing.

Orderly termination

An orderly termination of the network is invoked by the IMS master terminal operator using the IMS **/CHECKPOINT FREEZE**, **/CHECKPOINT PURGE**, or **/CHECKPOINT DUMPQ** command with the **QUIESCE** parameter.

QUIESCE is a **/CHECKPOINT** parameter that initiates shutdown processing for a network. When **QUIESCE** is specified, IMS sends the VTAM shutdown (SHUTD) command to all workstations and waits until all workstations have completed normal processing and returned the VTAM shutdown-complete (SHUTC) command.

When all workstations have indicated that shutdown is complete, IMS performs checkpoint processing and then issues the VTAM CLSDST macro instruction. CLSDST causes VTAM to send the **UNBIND** command to all workstations. This command releases the workstations from session with IMS. The controller prohibits any further data transmission from the workstations.

During shutdown processing (the time between the VTAM shutdown-complete and CLEAR commands), the system's workstations should be prepared for any IMS output that results from shutdown processing.

During the processing of an orderly termination, the IMS master terminal operator can terminate the network unconditionally rather than wait for the orderly termination processing to complete. This can be done by invoking an immediate termination.

Immediate termination

The IMS master terminal operator, the VTAM network operator, or the workstation user can invoke an immediate termination of a SLU P workstation.

The IMS master terminal operator uses the **/CHECKPOINT** command with the **FREEZE**, **PURGE**, or **DUMPQ** parameter, but without the **QUIESCE** parameter, to invoke immediate termination of the network. IMS issues the VTAM CLSDST macro instruction. CLSDST causes VTAM to issue the **UNBIND** command to all workstations. This command releases the workstation from session with IMS. The controller prohibits any further data transmission from the workstation.

To terminate workstations or portions of a network selectively, the IMS master terminal operator can use the IMS **/CLSDST** or **/STOP** command. **/CLSDST** and **/STOP** commands cause IMS to issue the VTAM CLSDST macro instruction for the specified workstations. The **/STOP** command also prevents further sessions from being established until a **/START** command is issued for the workstation.

The z/OS VTAM network operator uses the **VARY** command to terminate a workstation immediately. z/OS VTAM network operator intervention using the **VARY** command might be required to terminate sessions in which a workstation error prevents an I/O operation from completing.

The workstation uses VTAM session control commands to terminate a session with IMS. Under normal processing circumstances, when the workstation decides to terminate its session with IMS, that workstation should send the VTAM request-shutdown command. IMS completes any input or output currently in progress for that workstation and then issues the VTAM CLSDST macro instruction.

If the workstation detects an error condition from which it cannot recover and then wants to terminate the session, it can send the VTAM terminate-session command. VTAM releases the workstation from the session and notifies IMS accordingly.

Shutting down an IMS network (SLU P)

Shutting down an IMS network may or may not include IMS.

The IMS **/CHECKPOINT** command is used to invoke termination of the network and a shutdown of IMS. The format of **/CHECKPOINT** used determines whether the network termination occurs immediately or waits for workstation processing to complete:

/CHECKPOINT FREEZE|DUMPQ|PURGE QUIESCE
Allows all workstations to complete normal processing before shutting down IMS.

/CHECKPOINT FREEZE|DUMPQ|PURGE
Causes immediate session termination for all workstations.

Related tasks

“Immediate termination” on page 930
The IMS master terminal operator, the VTAM network operator, or the workstation user can invoke an immediate termination of a SLU P workstation.

SLU P messages

The primary function of IMS support for the SLU P system is to receive and transmit data for a workstation and to ensure that the data is processed properly.

Input/output messages can consist of the following data types:

- IMS transactions
- IMS message switches
- IMS commands
- VTAM commands and indicators
- Message Format Service (MFS) control requests

IMS can process nongraphic characters when sending and receiving any of these messages.

Related concepts

“IMS sensitivity to nongraphic message data” on page 437

The following topics describe the sensitivity IMS has to specific characters when users attempt to send and receive nongraphic data in IMS messages.

Send/receive and bracket protocol

The change-direction indicator and bracket protocol, as defined by VTAM, are used to specify the beginning and end of synchronous transmissions and to control the flow of such transmissions.

Because of its queued input and output processing, IMS does not support multiple related inputs or multiple related outputs. While in an in-brackets state, the following can occur:

- One input message (coded BB/EB)
- One output message (coded BB/EB)
- One input message and an output reply (input coded BB/CD and output coded EB)

The input message and output reply are related only for:

- Fast Path transactions
- Terminal-response mode transactions
- Conversational transactions

For other transaction types, the input message and output reply might or might not be related.

Related concepts

"Input bracketing protocol" on page 936
A workstation in session with IMS must indicate begin-bracket (BB) or both begin-bracket and end-bracket (EB) on both the first transmission of a chained message (multiple transmission) and each transmission of unchained messages.

"Output bracketing protocol" on page 942
For output messages, IMS specifies begin-bracket and end-bracket on single-segment output messages and on the first transmission of a multi-segment message.

Chapter 52. SLU P message protocols

A single transmission must be used for VTAM commands and indicators and MFS control requests. Single or multiple transmissions can be used to send IMS transactions, commands, and message switches.

Multiple transmissions for one message are required if the workstation's transmission buffer is not large enough to hold all of the data to be sent. Multiple transmissions can also be used if the workstation user wants to segment the data. Multiple transmissions are logically related through the concept of chained messages. Each transmission of a multiple transmission message is identified by its position in the chain—that is, first-in-chain, middle-in-chain, last-in-chain as shown in the figure in “Output messages” on page 936.

In a SLU P system using chained input messages, the chaining indicator is used to specify the chain position of the transmission; it should not be set for an unchained message. The chaining indicator must be turned on for the first-in-chain transmission, turned off for each middle-in-chain transmission, and turned on again for the last-in-chain transmission.

When MFS is defined, input messages can be processed by the MFS. For input formatting, MFS does not distinguish between possible input components but assumes that all input comes from the 4701/4702 application program.

General format of input function management headers (Finance)

IMS transactions, message switches, and commands are in the standard IMS format (transaction code, logical terminal name, or command verb, followed by a blank and text) unless MFS or a Physical Terminal Input edit routine is used to edit the message into the standard format.

The first or only transmission of IMS transactions can have a variable-length FM header. If MFS is not defined for the workstation, the header is ignored. If MFS is defined, the header can contain either an MFS control request or a MID name. For MFS control requests, only the first 2 bytes of the FM header are required.

Byte 0
Header length including the length byte (binary). Bytes 0 and 1 constitute an MFS control request.

Byte 1
Message description (binary).

Byte 2
MID name length (binary).

Bytes 3-10
MID name (1 to 8 bytes) (EBCDIC).

IMS uses the header, but removes it before sending the message to the MPP or MFS. When IMS receives a message with a header, it interrogates the length of the message. If the message length is 2 bytes, IMS assumes that the message is an MFS control request. The MFS control request is indicated in the message description (byte 1).

Related reading: For more information on using the PAGEREQ function, see *IMS Version 15 Application Programming APIs*.

Input message descriptor byte (Finance)

If the message length is more than 2 bytes, IMS interrogates the message description to determine whether an MFS MID name is present. The format description specified by the MID name is used to format the message.

The format of the message descriptor byte is:

Bit 0
Reserved

© Copyright IBM Corp. 1974, 2020

Bit 1
Page advance requested (NEXTPP)

Bit 2
Message advance requested (NEXTMSG)

Bit 3
Message advance protect requested (NEXTMSGP)

Bit 4
Next logical page requested (NEXTLP)

Bit 5
Reserved

Bit 6
Format description that is indicated by the MID name length and MID name fields should be used to format this input message.

Bit 7
Reserved

Bits 1, 2, 3, and 4 (MFS control requests) are mutually exclusive and are examined only if the message length is 2 bytes. Bit 6 must be set on if the input message is to be formatted by an MFS MID name. If bit 6 indicates an MFS MID name is present, bits 0 through 5 must be zero.

General format of input function management headers (SLU P)

IMS transactions, message switches, and commands are in the standard IMS format (transaction code, logical terminal name, or command verb, followed by a blank and text).

If a message is in nonstandard format, MFS or a Physical Terminal Input edit routine can be used to edit the message into the standard format. The first or only transmission of IMS transactions can have a variable-length FM header. If MFS is not defined for the terminal, the header is ignored. If MFS is defined, the header can contain either an MFS control request or optional MFS fields used to invoke MFS formatting. IMS uses the header in these cases, but removes it before passing the message on to the MPP or MFS. Incorrect header specifications or formats can cause the session to terminate. SLU P uses header type X'42'.

The format of the input FM header is:

Byte 0
Header length including the length byte in binary (first byte)

Byte 1
Header type (must be X'42')

Byte 2
Message descriptor 1 (flag byte is binary)

Byte 3
Message descriptor 2 (flag byte is binary)

Byte 4
Input component identification (binary)

Optional MFS fields:

For DPM:

Bytes 5 and 6
Version ID (in binary) if bit 0 of byte 3 is on.

Bytes 7-15
MID name length, including length byte (1 byte), followed by the MID name (1 to 8 bytes). If bit 0 of byte 3 is off (version ID not included in the FM header), the MID name length and the MID name are in bytes 5 through 13.

For SCS:

Bytes 3-13
MID name length (in binary), including length byte (1 byte), followed by the MID name (1 to 8 bytes).

When IMS receives a message with a header, it determines the length of the message. If the message length is 5 bytes, IMS assumes that the message is an MFS control request. The specific MFS control request is indicated in message descriptor 1 (byte 2). If the message length is more than 5 bytes, IMS examines the message description bytes to determine whether an MFS version ID and a MID name are present. When both are present, the version ID must immediately precede the MID name length. (Version ID does not exist for SCS1.) The format description specified by the MID name is used to format the accompanying input message, and the version ID is used to validate the format description level. If no FM header is provided, no version ID is provided, or a version ID of zero is provided, MFS bypasses the validity check on the format description level.

Related concepts
Version identification function for DPM formats (Application Programming APIs)

Input message descriptor bytes (SLU P)

Message descriptor 1 (byte 2 of the FM header) has the following format.

Bit 0
Reserved

Bit 1
Page advance requested (NEXTPP)

Bit 2
Message advance requested (NEXTMSG)

Bit 3
Message advance protect requested (NEXTMSGP)

Bit 4
Next logical page requested (NEXTLP)

Bit 5
Reserved

Bit 6
Format description indicated by the MID name length, and MID name fields should be used to format this input message

Bit 7
Reserved

Bits 1, 2, 3, and 4 (MFS control requests) are mutually exclusive. Bit 6 must be set on if the input message is to be formatted by the MFS MID name provided. When bit 6 indicates a MID name is present, bits 0 through 5 must be zero.

Message descriptor 2 (byte 3 of the FM header) has the following format:

Bit 0
Version identifier for MFS DPM. Bit 0 must be set on if bytes 5 and 6 of the FM header contain the version ID.

Bit 1-7
Reserved

Input component identification (SLU P)

The input component identification is used for the input component selected. The component identification can be a value between 0 and 4, matching the `ICOMPT` specified on the NAME macro during IMS system definition or on an ETO user descriptor.

If a component identification is zero, or if no FM header is sent, the input is associated with component 1. Further, IMS uses the input component identification to reset component output protection. If a value of 0

(zero) is sent, output protection is reset on all components of the terminal. If a value 1 through 4 is sent, output protection is reset on only the specified component.

Related concepts

"Component definition" on page 907
IMS considers a *workstation* to be one physical terminal. If a workstation is made up of more than one device, each physical device that makes up the workstation must be defined as a component of that workstation.

Related tasks

"Administering the Extended Terminal Option" on page 69
The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS without having to first define them during system definition.

Input bracketing protocol

A workstation in session with IMS must indicate begin-bracket (BB) or both begin-bracket and end-bracket (EB) on both the first transmission of a chained message (multiple transmission) and each transmission of unchained messages.

The end-bracket indicator places both IMS and the workstation between brackets and in a contention state. If only the begin-bracket indicator is sent, the workstation should send a change-direction (CD) indicator on each of the following:

- The last transmission of a chained message (multiple transmission)
- Each transmission of unchained messages

The CD indicator is necessary, because IMS cannot correlate multiple input messages. The CD indicator is implied if it is not specified.

If BB and CD are indicated, IMS indicates EB on its next output to the workstation. If IMS detects an error while receiving chained input on which only BB is specified, it returns an EB and an IMS error message.

Restriction: For SLU P terminals, a CD or EB indicator must accompany each input chain; otherwise, the session terminates.

Related concepts

"Output messages" on page 936
Output messages from IMS can be one of several different types.

Related reference

"Error handling" on page 948
The following topics describe the procedures used by IMS and required of the controller or the controller application program to handle failures resulting from transmission or protocol errors.

Activating MFS input formatting for Finance workstations

When MFS is used, input messages can be processed by the message and format descriptors.

When IMS receives an input message, IMS Basic Edit is performed unless a MID name accompanies the message. The MID name can be supplied by including it either in the input function management header or in the beginning of the message text with the required MFS escape characters (//). When the MID name is present, MFS edits the message using the specified MID and its associated device input format (DIF). The MID name is provided by either the operator or the remote application program.

Output messages

Output messages from IMS can be one of several different types.

The different types of output messages from IMS include:

- Data replies to recoverable or irrecoverable input transactions
- Data replies to IMS commands
- Message switches
- VTAM indicators
- IMS system messages
- Broadcast messages
- A null (length=0) message containing end-bracket to return the workstation to between-brackets and to a contention state

All messages from IMS to a workstation are sent in a single transmission unless:

- The MPP, a message switch, a command, or MFS provides a multi-segment output message.
- The workstation's read buffer is too small to hold the single-segment output message provided by the MPP or MFS.
- A broadcast message is multi-segment.

Each segment of a message is sent in a single transmission whenever possible, that is, when only-in-chain is indicated. However, if a message segment exceeds the size of the receiving workstation read buffer, the segment is divided into as many transmissions as required until the complete segment is sent. Multi-segment output messages are handled like chained input messages: Each segment is identified appropriately as first-in-chain, middle-in-chain, or last-in-chain.

Exercise care when defining the sizes for IMS message queue data sets and workstation output buffers. Incorrect specification can result in multi-segment output chains from IMS. Output buffer sizes that are too small can result in multiple transmissions for large IMS segments. Message queue data set sizes that are too small can result in multiple IMS segments being created for large message processing program (MPP) inserts.

Related reading: For information on the MSGQUEUE and TERMINAL macros, see *IMS Version 15 System Definition*.

Definition: *MFS-paged output* is each physical page of a message destined for a display device and sent as if it were a complete message.

The following figure shows the relationship between transmissions sent to IMS, segments produced by the MPP, and segments transmitted by IMS.

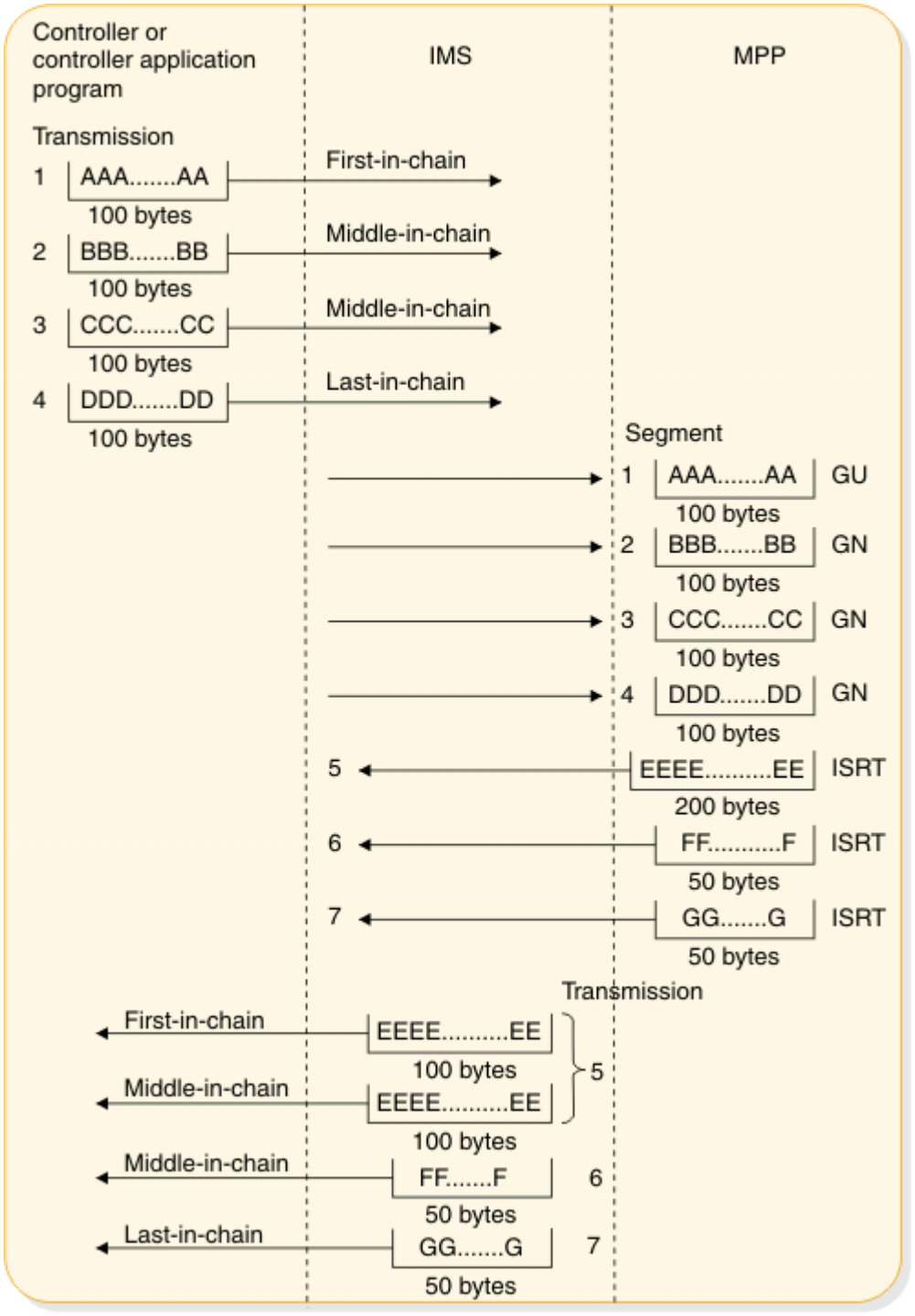


Figure 159. Chained message interaction between a Finance Communication system and an IMS MPP

After successfully receiving a message, the SLU P system must interrogate the read-type field (SMSCRT), the read-flags field (SMSCRF), and, optionally, the read-flags field extension (SMSCRE) to determine the type and characteristics of the message received.

Related reading: For more information on these fields, see *IBM 4700 Finance Communication System: Controller Programming Library, Volume 3: Communication Program*.

If a non-MFS-formatted output segment is sent in multiple transmissions and the SCAN option is defined for that workstation, IMS does not allow a 4701/4702 device's select control, position-control, or write-control sequence to be split across transmissions. When transparent write-control sequences are split

into two transmissions, the data length on the first transmission is modified, and the proper write control characters are inserted at the beginning of the second transmission.

Editing for device control sequences does not occur for non-MFS-formatted output if the NOSCAN option is specified during IMS system definition or on an ETO logon descriptor. If MFS formats the output segment, the device-control sequence (select and position) editing always occurs. Each transmission of the segment, except the last, ends in a three-character, null-horizontal-position sequence to suppress the automatic new-line function of the physical SLU P system output device.

MFS Distributed Presentation Management output (SLU P)

IMS sends the entire output message as a single chain of one or more related transmissions. For MFS DPM output using a device format defined with paging OPTIONS=DPAGE or OPTIONS=PPAGE, IMS sends each logical or presentation page in a single chain of one or more related transmissions as if it were a complete message.

For MFS DPM output using a device format defined with paging OPTIONS=DPAGE or OPTIONS=PPAGE, IMS sends an output function management header as part of the first or only transmission of each chain (logical or presentation page) of the message. The remainder of the transmission contains data fields (DFLDs) up to the defined record length. This function management header contains the DPM version ID, DPAGE or PPAGE name, and a MID name if specified.

When a forms literal is defined in the DPM format specification for paging OPTIONS=DPAGE or OPTIONS=PPAGE (FORMS= parameter on the DEV statement), IMS precedes the first logical or presentation page with an only-in-chain transmission consisting of a function management header containing only the forms literal. No version ID, DPAGE name, or PPAGE name is provided. Output components are protected at the end of each output chain; this prevents IMS from sending subsequent output, or logical or presentation pages (chains), until an appropriate input message or control request is received.

For MFS DPM output using a device format defined with paging OPTIONS=MSG, IMS sends the entire message as a single chain of one or more related transmissions. The function management header is sent as the first or only transmission and contains the DPM version ID, format name, and any MID name or forms literal defined in the MFS format specification. In this case, as for non-DPM output, component protection is provided only if the destination component is defined as PROGRAM2.

For DPM, the device type symbolic name is specified to MFS as DPM-Xn, where X can be A or B. Message formatting is specified on the TERMINAL macro or on an ETO logon descriptor for logical units defined as SLU P by using device type symbolic names of the form DPM-An. DPM-Bn device type symbolic names refer to logical units defined for Intersystem Communication.

General format of output function management headers (Finance)

IMS appends a header to the first or only-in-chain transmission of all messages sent, except the IMS null message and VTAM commands and indicators.

This header describes the type of message that follows and, if appropriate, indicates the output component that is to receive the message. The header can contain optional MFS information.

The length of the output function management header varies from 3 to 29 bytes. The format is:

Byte 0
Header length including length byte (binary)

Byte 1
Message description (binary)

Byte 2
Output component identification (binary)

Bytes 3-28
MFS data (length byte: binary; names: EBCDIC)

Output message descriptor byte (Finance)

The bits of the message descriptor byte (byte 1), when set on, have the following meanings.

Bit 0
This message is an IMS system message or a broadcast message.

Bit 1
This message is formatted by MFS.

Bit 2
An MFS system control area (SCA) has requested device alarm for this message. (The controller application program can take any appropriate action, such as turning on a terminal's light.)

Bit 3
This message is a nonqueued message (no specific destination). Bit 0 is also set.

Bit 4
Reserved.

Bit 5
This message is Fast Path-recoverable output.

Bit 6
A MID name is present in the MFS data field.

Bit 7
A FORMS name is present in the MFS data field.

Output component ID byte (Finance)

The output component identification (byte 2) contains a value from X'01' to X'04'.

This value identifies the terminal component to which this message is directed. Output component identifiers are assigned during IMS system definition, on an ETO user descriptor, or by using the Signon (DFSSGNX0) and the Output Creation (DFSINSX0) exit routines. These identifiers are specified by the sender of the message (either IMS or the message processing program).

Related concepts

"Output component selection" on page 907
IMS system definition allows a workstation to have a maximum of four output components.

MFS data bytes (Finance)

Bytes 3 through the end of the header can contain MFS data, and one or two fields can be present. Bits 6 and 7 of the message descriptor byte indicate whether these fields are included.

The first field, if present, is 2 to 9 bytes long and contains a 1-byte length indicator including length byte and a 1- to 8-byte MFS MID name. This MID name field is present if the message output description contains the name of the MID to be used for the next input message, and should be returned to IMS by the controller program. The second field, if present, is 2 to 17 bytes long and contains a 1-byte length indicator including length byte and a 1- to 16-byte FORMS name. FORMS name identifies the special form required for this message. Before printing the message, the application program should ensure that the form is in place and that page size and forms alignment are established.

General format of output function management headers (SLU P)

IMS appends a header to the first or only-in-chain transmission of all messages sent, except the IMS null message and VTAM commands and indicators.

This header describes the type of message that follows and, if appropriate, indicates the output component that is to receive the message. The header can also contain optional MFS information. SLU P uses header type X'42'.

The length of the output function management header varies from 5 to 42 bytes. The format is:

Byte 0
Message header length including length byte (binary)

Byte 1
Header type (X'42')

Byte 2
Message descriptor 1 (flag byte is binary)

Byte 3
Message descriptor 2 (flag byte is binary)

Byte 4
Output component identification (binary)

Bytes 5-41
MFS data (length byte: binary; names: EBCDIC)

Output message descriptor bytes (SLU P)

The bits of the message descriptor 1 (byte 2) have the following meaning when set on.

Bit 0
This message is an IMS system message or a broadcast message.

Bit 1
This message is formatted by MFS.

Bit 2
Reserved.

Bit 3
This message is a nonqueued message (no specific destination). Bit 0 is also set.

Bit 4
Reserved.

Bit 5
Fast Path-recoverable output.

Bit 6
A MID name is present in the MFS data field.

Bit 7
A forms name is present in the MFS data field.

The bits of message descriptor 2 (byte 3), when set on, have the following meaning:

Bit 0
This output is formatted by MFS DPM. The version ID might or might not be present, and one of the following is present in the MFS data field:

- DPM format name
- Logical page name (DPAGE statement label)
- Presentation name (PPAGE statement label)

Bit 1
Reserved.

Bit 2
At the end of this chain, the component indicated in byte 4 of the output function management header is protected.

Bits 3-7
Reserved.

If message descriptor 2, bit 2, is set on, IMS does not send another output chain until an appropriate input occurs. This input can be an input message, an MFS control request, or a `READY-TO-RECEIVE` (RTR) command.

Related concepts

"Extended output component protection (SLU P)" on page 914
When a SLU P is defined with the NOBID option, IMS provides support similar to the screen-protection function for the IBM 3270 Information Display System and display-screen protection for Finance components.

MFS data bytes (SLU P)

Bytes 5 through the end of the header can contain up to four MFS fields, depending on settings in message descriptor 1, bits 6 and 7, and message descriptor 2, bit 0.

If bit 0 is on, the first field, if present, is a 2-byte MFS DPM version identification. This field is present for all MFS DPM formatted output. The field can contain a version ID, or the value 0 (zero). In either case, one of the following fields (second, third, or fourth field) is present.

The second field, if present, is 2 to 9 bytes long and contains a 1-byte length indicator, including length byte followed by a 1- to 8-byte MFS MID name. This MID name field is present if the message output description contains the name of the MID to be used for the next input message, and should be returned to IMS by the controller program.

The third field, if present, is 2 to 9 bytes of data name, as defined by the paging option for the MFS device format.

The fourth field, if present, is 2 to 17 bytes long and contains a 1-byte length indicator, including length byte followed by a 1- to 16-byte user-specified MFS forms literal. The forms literal identifies the special setup or forms required for this message. The user must define the procedure to be followed at the terminal upon receipt of the forms literal.

The message format name (FMT statement label) is present if OPTIONS=MSG is specified; the logical page name (DPAGE statement label) is present if OPTIONS=DPAGE is specified; the presentation page name (PPAGE statement label) is present if OPTIONS=PPAGE is specified. This field appears in all output (message descriptor 2, bit 0) from DPM.

The content and format of the function management header for DPM formatted output can be influenced when defining the MFS device format through the HDRCTL=FIXED or HDRCTL=VARIABLE option. If HDRCTL=VARIABLE is specified, each MFS header field is variable in size. If HDRCTL=FIXED is specified, the following MFS header fields are sent and padded to their maximum size.

- The MID name field in the output header is padded with trailing blanks for the maximum length of 8 bytes. Eight blanks are sent if no MID name is specified to format the next input through the user-supplied message output description.
- The message format name is padded to a maximum length of 6 bytes. The logical page name or the presentation page name is padded to a maximum length of 8 bytes.

Output bracketing protocol

For output messages, IMS specifies begin-bracket and end-bracket on single-segment output messages and on the first transmission of a multi-segment message.

If the input message specified begin-bracket and change-direction, the next output IMS sends specifies only end-bracket. If IMS detects an error when receiving chained input on which only begin-bracket is specified, it returns end-bracket and an IMS error message.

In the case of Fast Path, conversational, or response-mode transactions, the output message is the reply to the previous input message. In other cases, the output message might or might not be related to the previous input message.

IMS sends an only-in-chain output message (no FM header and data length=0), requesting exception DR2, and indicating FM header end-bracket when no other output is immediately available. This occurs in the following situations:

- An input message from a workstation defined to IMS as "negated terminal-response mode" specifies begin-bracket and change-direction.

- An input message from a workstation defined to IMS as "forced terminal-response mode" specifies begin-bracket and change-direction and is not an IMS transaction.
- An input message from a workstation defined to IMS as "transaction-dependent terminal-response mode" specifies begin-bracket and change-direction and is not a response-mode transaction.

After receiving exception-response sense codes 0811 and 0812, IMS also sends an only-in-chain output message (no FM header and data length=0), requesting exception DR2, and indicating FM header begin-bracket/end-bracket when no other output is immediately available.

Related reference

"Error handling" on page 948
The following topics describe the procedures used by IMS and required of the controller or the controller application program to handle failures resulting from transmission or protocol errors.

Activating MFS output formatting for SLU P

MFS output formatting occurs when an output message has an associated message output descriptor (MOD).

An output message has an associated MOD when any of the following occurs:

- The MPP supplies the name of a MOD (MOD name) with the output message.
- The input message was processed by a message input descriptor (MID) whose definition specified a MOD name for output formatting.
- The output message is a message switch from a device using MFS editing.

If no MOD is associated with the output message, standard IMS output editing occurs.

Input formatting for a workstation only occurs when a MID name and the message itself are sent to IMS. Specifying the next MID name is allowed in the message output description. Controller involvement is required to assure the proper MID is used, because IMS cannot directly control the receipt of input. Therefore, while IMS is formatting an output message, the workstation can be sending an input message. Input sent while IMS is processing output is queued by VTAM until IMS completes output processing. If the next MID name is specified, the input queued by VTAM appears as if it were the result of the current output. Unpredictable results occur if the internal MID name is used to format the message. IMS avoids this by sending the MID name as part of the input message.

IMS informs the controller that the name of the next MID is specified, and sends the MID name requested and the output message to the workstation. The controller then saves the MID name, displays the output message, reads the next operator input, adds the saved MID name to the transaction, and sends the transaction to IMS.

Response requests (Finance)

Output messages from IMS require specific responses. IMS commands, broadcast output, message switches, and non-Fast Path transactions defined to IMS as update-inquiry or recoverable-inquiry require a DR2, except replies to all but the last physical or logical page of non-operator, logical-paged MFS output.

Transactions defined to IMS as irrecoverable-inquiry or replies to operator logical-paged MFS output require an exception DR2. Except for the last physical page of a message whose MOD does not specify PAGE=YES, MFS-paged output requests an exception DR2 reply (regardless of how the transaction is defined), because an explicit MFS control or data input must follow each page before another page of output is allowed; otherwise, the message is dequeued.

Fast Path output messages are sent requesting an exception DR2 response regardless of the transaction type. The next input message from the workstation or an RTR command provides acknowledgment that the preceding output has been received and that IMS might dequeue the message.

Related reference

"Input response requirements" on page 944

The following table summarizes input requirements for requesting responses for all types of data transmission between a workstation and IMS.

Response requests (SLU P)

Except for replies to all but the last physical or logical page of non-operator, logical-paged MFS output, output messages from IMS require specific responses. IMS commands, broadcast output, message switches, and non-Fast Path transactions defined to IMS as update-inquiry or recoverable-inquiry require a DR2.

Because an explicit MFS control or data input must follow each page before another page of output is allowed (or the message is dequeued), transactions defined to IMS as irrecoverable-inquiry and replies to operator logical-paged MFS output require an exception DR2. MFS-paged output, except the last physical page of a message whose MOD does not specify PAGE=YES, requests an exception DR2 reply regardless of how the transaction is defined.

Related reference

"Input response requirements" on page 944
The following table summarizes input requirements for requesting responses for all types of data transmission between a workstation and IMS.

Input response requirements

The following table summarizes input requirements for requesting responses for all types of data transmission between a workstation and IMS.

Table 179. Input response requirements by message type

Data type	Responses accepted (ACK option)	Responses accepted (OPTACK option)
Update transaction	DR2"1" on page 945 Exception DR1 or exception DR2"3" on page 945	DR1 or DR2 Exception DR1 or exception DR2"1" on page 945, "2" on page 945
Recoverable-inquiry transaction	DR2"1" on page 945 Exception DR1 or exception DR2"3" on page 945	DR1 or DR2 Exception DR1 or exception DR2"1" on page 945, "2" on page 945
Irrecoverable-inquiry transaction	DR1 or DR2 Exception DR1 or DR2"1" on page 945	DR1 or DR2 Exception DR1 or exception DR2"1" on page 945, "2" on page 945
Fast Path transaction"4" on page 945	N/A	DR1 or DR2 Exception DR1 or exception DR2"1" on page 945, "2" on page 945
IMS message switch	DR2"1" on page 945	DR1 or DR2 Exception DR1 or exception DR2"1" on page 945, "2" on page 945
IMS command	DR1 or DR2 Exception DR1 or DR2"1" on page 945	DR1 or DR2 Exception DR1 or exception DR2"1" on page 945, "5" on page 945
VTAM command or indicator	DR1"1" on page 945	DR1"1" on page 945
SLU P system MFS control request	DR1 or DR2 Exception DR1 or DR2"1" on page 945	DR1 or DR2 Exception DR1 or DR2"1" on page 945

Notes:

1. Recommended or required response.
2. Fast Path users should always use OPTACK option with exception DRx.
3. With CD only for the commands /DIS, /RDIS, and /FOR.
4. ETO users should always use OPTACK option.
5. With Change Direction (CD) only.

Output response requirements

The following table summarizes the response to be requested by IMS for all types of output messages.

Table 180. Output response requested by message type

Output data type	Response requested
Update, Recovery	DR2
Inquiry, Recovery	DR2
Inquiry, Irrecovery	Exception DR1 and DR1"3" on page 945
Fast Path (Recovery)	Exception DR2"1" on page 945 and DR2"2" on page 945
Last MFS Page (when PAGE=YES is not specified on the MOD)	Refer to above output data types for response request
Nonlast MFS Page (or all pages when PAGE=YES is specified on the MOD)	Exception DR2 and DR2"2" on page 945
IMS Command Replies: /FORMAT, /DISPLAY, /RDISPLAY Other Commands	DR2 Exception DR2 and DR2"2" on page 945
Test Mode Output	Exception DR1 and DR1"3" on page 945
Broadcast or Message Switch Output	DR2

Notes:

1. Next input message or RTR provides acknowledgment if exception DR2 is requested.
2. DR2 is requested when one of the following occurs:
   - The message is sent with BB (bidding with data).
   - Messages on the IMS message queues are waiting to be sent to the workstation.
   - The NFPACK option is defined on the system definition TERMINAL macro or on an ETO logon descriptor.
3. DR1 is requested when the message is sent with BB (bidding with data).

IMS transaction types

Transactions are the most common data type sent from a workstation to IMS. IMS supports two kinds of transactions—update and inquiry.

An update transaction can modify a database. An inquiry transaction can look at data in a database but cannot change or update it. Transactions are defined as update or inquiry during IMS system definition.

An additional attribute is defined for inquiry transactions—recoverable or irrecoverable. Recoverable-inquiry transactions are always recoverable no matter which element in the network fails. Irrecoverable-inquiry transactions are not recovered following an I/O error condition or IMS system restart.

All update transactions are recoverable.

All Fast Path transactions must be defined as recoverable, but can be either inquiry or update.

A workstation can be defined to handle one or both types of transactions. Other decisions and processing might be required of the controller or controller application program when both recoverable and irrecoverable transactions are handled. The decision to define a transaction as recoverable or irrecoverable requires an evaluation of the advantages and disadvantages that each transaction type offers to the individual operating environment.

The following topics describe the inquiry message types and the responses required for each.

Recoverable-inquiry transactions

To ensure that a recoverable transaction can be recovered, the workstation must perform certain required functions.

To ensure that a recoverable transaction can be recovered, the workstation must:

- Request a DR1 or DR2 on input to IMS, or, if the OPTACK option is defined, optionally request an exception DR1 or exception DR2 and specify begin-bracket and change-direction on input to IMS.
- Maintain the input message sequence number, and, optionally, a copy of the input message until the DR1 or DR2 is returned, or, if the OPTACK option is specified, until a reply message containing an end-bracket is returned.
- Return any DR2 requested by IMS, or, if Fast Path, ensure that another input message or RTR command is sent after having accepted responsibility for an IMS output message by either logging or writing out the data.

To ensure that a recoverable transaction can be recovered, IMS:

- Requests a DR2 response, or, if Fast Path output and no output messages are waiting to be sent, requests exception DR2 and waits for subsequent data or RTR as acknowledgment of the output.
- Maintains the message sequence number and a copy of the message until a DR2, input data, or RTR, if Fast Path, is returned.
- Returns any requested DR1 or DR2 after having accepted responsibility for the message by either logging or writing out the data.

If a failure occurs between the sending of a recoverable message and receipt of the acknowledgment or reply message, the sender cannot determine whether the message reached its destination. During the restart procedure, IMS uses the VTAM STSN command to inform the controller of the sequence numbers of the last inbound sync-point message IMS received and of the last outbound sync-point message IMS sent. Any messages that have not been received can then be retransmitted.

When recoverable messages are being sent, only one recoverable message can be outstanding at a time. This means that the sender should send one message and wait for the response or reply before sending another. IMS reads a message, places the message on the input queue, and returns the DR1 or DR2 response or reply before it accepts another message.

Irrecoverable-inquiry transactions

IMS treats an irrecoverable transaction in the same manner as a recoverable transaction, except that all processing required to achieve recoverability is eliminated. As a result, irrecoverable transactions require less processing time but can be lost in the event of a failure (for example, line failure, processor failure, queue failure) in the network.

An irrecoverable transaction need not request a DR1 or DR2. Because recoverability is not guaranteed, the receiver of the message is not required to acknowledge receipt. An irrecoverable transaction should, however, request an exception DR1 or DR2. The exception DR1 or DR2 requires fewer line transmissions than DR1 or DR2, because under normal circumstances no response is returned.

Verifying IMS receipt of irrecoverable messages

The SLU P system does not request that IMS acknowledge receipt of irrecoverable messages. However, if the system is transmitting multiple irrecoverable messages, indicating both begin-bracket and end-bracket, and wants to verify their receipt, it can send the VTAM CHASE command.

VTAM indicators must request a DR1. When the SLU P system receives the normal definite response 1, all messages preceding the CHASE command have been received by IMS.

IMS message switches

IMS treats a message switch that is sent or received just like a recoverable transaction.

For this reason, the following applies to message switches:

- A message switch must request a DR1 or DR2 if the workstation is defined with the ACK option. If the OPTACK option is specified, a message switch optionally requests an exception DR1 or DR2 and should specify begin-bracket and change-direction on input to IMS.
- A message switch is recovered by IMS.
- The sequence number of a message switch is used in message resynchronization.

IMS commands

The network system analyst decides whether a workstation is allowed to enter IMS commands and, if so, which commands are allowed.

IMS does not require that the controller request a specific response when sending an IMS command. DR1, DR2, and exceptions DR1 and DR2 are allowed; exception DR1 or DR2 is recommended.

IMS processes its commands and returns an IMS message when it finishes processing the command. Therefore, when a DR1 or DR2 is requested, IMS returns the DR1 or DR2, followed immediately by the message confirming command completion.

The additional line transmission required to send either the DR1 or DR2 and the command completion message can be eliminated if the controller requests exception DR1 or DR2.

If exception DR1 or DR2 is requested, the command completion message acknowledges receipt of the command and indicates that command processing is complete. If the ACK option is defined for the workstation, the VTAM sequence number of the command does not participate in message resynchronization as it does when only DR1 or DR2 is requested. If the OPTACK option is defined, all input participates in message resynchronization.

VTAM commands and indicators

When a VTAM command is sent to IMS, a DR1 must be requested.

When IMS sends a VTAM command, it requests a DR1. If the BIND or BID command is sent, the response must be either a DR1 or exception DR1. If the STSN command is sent, a DR1 is required.

The controller responds to all other VTAM commands and indicators for the workstation.

MFS control requests

When Message Format Service (MFS) is used, MFS control requests can be used to display paged messages and to control the display component screen.

The control request is specified in the input function management header.

DR1, DR2, or exception DR1 and DR2 can be requested. Using exception DR1 or DR2 is recommended, because the additional line transmission required to send DR1 or DR2 and the completion message can be eliminated if the controller requests exception DR1 or DR2.

Error handling

The following topics describe the procedures used by IMS and required of the controller or the controller application program to handle failures resulting from transmission or protocol errors.

IMS-detected errors

Whenever IMS detects an abnormal send or receive condition from VTAM, NCP, or the controller, it terminates the session. When an error on a received message is detected, IMS protects any display component and returns an exception DR1 or DR2 that includes 4 bytes of sense information.

Bytes 0, 1
System sense field

Bytes 2, 3
User sense field

System sense field (SSENSE)

Bytes 0 and 1 contain one of the following values.

X'0800'
The user sense field contains the user message from the user message table.

X'0819'
The user sense field contains the IMS message number 290 (in binary). No output is available. This value is set only for a non-MFS response to a previously received RTR command.

X'0826'
The user sense field contains the IMS message number.

User sense field (USENSE)

IMS uses the user sense field to pass on the number (converted to binary) of the appropriate error message.

For example, an invalid transaction code generates the message:

```
DFS064  DESTINATION CANNOT BE FOUND OR CREATED
```

If IMS receives an invalid transaction code, it returns an exception DR1 or DR2 response with X'0040' in the user sense field.

In the XRF complex, the alternate system sends SSENSE=X'0826', USENSE X'0F15', and a DFS3861I SYSTEM TAKEOVER OCCURRED message if an input transaction is lost across the XRF session takeover. You must then retransmit the last input record.

IMS binds each session with unconditional bracket termination. When IMS sends an exception DR1 or DR2 to input, it must remain in a send state (regardless of the current bracket state), and the workstation must go to a receive state. Both IMS and the workstation must go to a between-brackets state if the input message in error specified end-bracket. IMS then places the workstation between-brackets and into a contention state by sending the entire IMS error message as a single chain with the appropriate bracket indicators.

An IMS error message can have multiple segments, with a maximum message length of 32,000 bytes. However, all current IMS error messages that are sent to the workstation are single-segment and less than 132 bytes. These messages adhere to the FM header and transmission rules for all output messages.

The generation of the user sense data is dependent on the standard IMS error message format. The session can be terminated if any user Output edit routine modifies this format.

Related concepts
“Considerations for controller application programs for XRF systems” on page 901

If your SLU P system operates in an XRF complex, your controller application program must be able to handle messages lost during an XRF takeover.

Controller or station-detected errors

Whenever the controller detects an error on a message from IMS, or simply cannot accept the message at that time, an exception DR2 that includes 4 bytes of sense data is returned.

Bytes 0, 1
System sense field

Bytes 2, 3
User sense field

System sense field

Bytes 0 and 1 should contain either X'0802' (RECOVERABLE ERROR) or X'0811' (BREAK).

X'0802'
RECOVERABLE ERROR—If the message is recoverable, IMS returns it to the output queue and retransmits it following the next input from that workstation or whenever additional output for that destination is queued. Retransmission of chained messages begins from the first-in-chain transmission. Because any open bracket is closed upon receipt of the exception DR2, the retransmitted message contains end-bracket only or both begin-bracket and end-bracket indicators (depending upon whether input occurred or whether the bracket indicators were specified). Irrecoverable messages have been dequeued and cannot be recovered and retransmitted.

X'0811'
BREAK—Cancel output. The output message is terminated. This message is dequeued from IMS and is not retransmitted; it is not subject to recovery. Any remaining output messages that are queued for the workstation are sent. If additional output is not available after receiving this sense code, IMS sends a null output message.

X'0812'
TEMPORARY ERROR—Resource not available. IMS action same as for X'0802'.

X'0813'
TEMPORARY ERROR, BID OR BRACKET REJECTED—NOT READY TO RECEIVE follows. IMS action is the same as for X'0802'.

X'0814'
TEMPORARY ERROR, BID OR BRACKET REJECTED—READY TO RECEIVE follows. IMS action is the same as for X'0802'.

X'081C'
STATION/COMPONENT DOWN—If byte 3 indicates a valid component (X'01' through X'04'), IMS marks the indicated component as inoperable. If byte 3 indicates X'00' or an invalid component (other than X'01' through X'04'), IMS interprets this as a session termination request and terminates the session. If the message in progress is recoverable, IMS returns it to the output queue and retransmits it when the workstation or component is again operable. Irrecoverable messages cannot be recovered and retransmitted.

Any other value for bytes 0 and 1 are handled as a request to terminate the session.

If the BID option is defined, the controller application program should use either X'0813' or X'0814' in an exception DR1 to the BID command; otherwise, the session is terminated because of an invalid response.

The X'0813' sense code results in the message being returned to the IMS output queue if it is recoverable, or dequeued if it is irrecoverable. Any recoverable data returned to the queue causes the bid to be sent again following the next input from the workstation, or whenever additional output for that destination is queued.

The X'0814' sense code causes a recoverable message to be returned to the IMS output queue, or a irrecoverable message to be dequeued. IMS then waits for an RTR command. The workstation can now send input to IMS. Until an RTR is received, IMS does not initiate any output when in a between-brackets

state. If IMS is left in an in-bracket send state (the last input carried BB/CD), IMS sends output with EB to terminate the current bracket.

Related concepts

“Output messages” on page 936
Output messages from IMS can be one of several different types.

User sense field

This field is used when the system sense field is set to X'081C'. Byte 2 is not inspected by IMS and, therefore, can contain any value.

Byte 3 contains the hexadecimal identification (X'nn') of the component to which the output message is directed. IMS marks that component as inoperable and continues to send output to other operable components of the workstation. If X'00' or an invalid component is specified, the session is terminated.

IMS binds each session with unconditional bracket termination. This means that both IMS and the workstation go to between-brackets and contention states for all supported sense codes except X'0813' and X'0814'. For these two sense codes, IMS goes to between-brackets and receive states; the workstation goes to between-brackets and send states.

VTAM logical unit status (LUSTATUS) command

If the SLU P system detects a component-down or workstation-down condition, and the SLU P system is not currently receiving a message from IMS, it can send the VTAM **LUSTATUS** command to notify IMS of the condition.

Four bytes of data, identical in format to the exception DR1 or DR2 for `STATION/COMPONENT DOWN`, must accompany the **LUSTATUS**. When IMS receives the **LUSTATUS**, it marks the indicated component inoperable or terminates the session.

The workstation can also notify IMS when the component is operable by sending the VTAM **LUSTATUS**. The system sense field must equal X'0001', and the user sense field byte 3 must equal a value from X'01' through X'04'. IMS resets the inoperable condition for the indicated component and sends any output queued for it.

All other values for **LUSTATUS** are considered requests to terminate the session.

VTAM ready-to-receive (RTR) command

The following list summarizes the previously defined IMS functions performed upon receipt of an RTR command.

- Any outstanding Fast Path output message is dequeued and IMS removes the workstation from terminal response mode.
- Any output-protected components are marked unprotected and available for output.
- Output that is suspended because of a X'0814' sense code on a previous exception response is now allowed.
- If IMS output is available to be sent, IMS returns a DR1 response, immediately followed by the output message.
- If no IMS output is available, or if the workstation is in a status where no output can be sent (output stopped or quiesced), IMS returns an exception DR1 followed by an error message indicating no output is available.
- If an RTR is received during the transmission of MFS-paged output, IMS returns an exception DR1 followed by an error message indicating invalid paging request.
- Any exception response returned by IMS causes any display component defined for the logical unit to be protected and unavailable for output.

VTAM CANCEL command

The VTAM **CANCEL** command is used by IMS, and should be used by the SLU P system to terminate a chained message in progress that has not completed.

IMS sends the **CANCEL** command to a workstation in these situations:

- The IMS master terminal operator issues the IMS **/DEQUEUE** command and IMS has not yet sent the last-in-chain transmission of a message.
- IMS receives an exception DR1 or DR2 to a transmission of a message and has not yet sent the last-in-chain transmission.

The controller or controller application program should use the **CANCEL** command in these situations:

- When an exception DR1 or DR2 is received from IMS prior to sending the last-in-chain transmission of a message.
- When multi-RU chained input to IMS must be canceled prior to sending the last-in-chain transmission of a message. In this case, if the message to be canceled contains only the begin-bracket indicator, the VTAM **CANCEL** command can include the end-bracket indicator, the change-direction indicator, or no additional indicators.

VTAM request-recovery command

If, during its normal operation, a workstation detects an error condition that does not terminate the session with IMS but does cause the workstation to resynchronize with IMS (for example, diskette failure in the controller or a program check), the application program can initiate message resynchronization.

Initiation of message resynchronization is done by sending the VTAM request-recovery (RQR) command. IMS responds with a **CLEAR** command followed by the set-and-test-sequence-numbers (STSN) command. Message resynchronization follows and is the same as that described under "Message recovery" on page 918.

Restriction: Session initiation and resynchronization caused by an SNA Request-Recovery (RQR) command is not allowed when the node is in response mode and the response reply message is not yet available for output; that is, the input response mode transaction is still queued or in the process of execution. Response-mode transactions are not recoverable or restartable prior to the application sync point; therefore, session input acknowledgment does not occur until input processing is complete.

Four bytes of data can accompany a SIGNAL command that is sent to IMS. The first two bytes are system-signal data. The last two bytes are user data that is ignored by IMS. The first two bytes are handled as follows, with all other values being ignored:

X'0000'
No specific action is taken. This data can, however, cause IMS to send any available output for operative, unprotected components if the workstation is idle due to a previous exception DR1 or DR2 sent to IMS. This input does not reset the output component protection or display protection.

X'0001'
This data is treated as an attention signal that causes IMS to stop sending and to wait for input at the end of the current output message. This type of signal can be used before sending either input or required DR1 or DR2 or exception DR1 or DR2 responses to IMS.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan, Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and

© Copyright IBM Corp. 1974, 2020

cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows: © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

Programming interface information

This information documents Product-sensitive Programming Interface and Associated Guidance Information provided by IMS, as well as Diagnosis, Modification or Tuning Information provided by IMS.

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service. Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a section or topic, or by a Product-sensitive programming interface label. IBM requires that the preceding statement, and any statement in this information that refers to the preceding statement, be included in any whole or partial copy made of the information described by such a statement.

Diagnosis, Modification or Tuning information is provided to help you diagnose, modify, or tune IMS. Do not use this Diagnosis, Modification or Tuning information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs, either by an introductory statement to a section or topic, or by the following marking: Diagnosis, Modification or Tuning Information.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek

your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

To learn more, see IBM Privacy Statement.

Bibliography

This bibliography lists all of the publications in the IMS 15 library.

Title	Acronym	Order number
IMS Version 15 Application Programming	APG	SC27-6778
IMS Version 15 Application Programming APIs	APR	SC27-6779
IMS Version 15 Commands, Volume 1: IMS Commands A-M	CR1	SC27-6780
IMS Version 15 Commands, Volume 2: IMS Commands N-V	CR2	SC27-6781
IMS Version 15 Commands, Volume 3: IMS Component and z/OS Commands	CR3	SC27-6782
IMS Version 15 Communications and Connections	CCG	SC27-6783
IMS Version 15 Database Administration	DAG	SC27-6784
IMS Version 15 Database Utilities	DUR	SC27-6785
IMS Version 15 Diagnosis	DGR	GC27-6786
IMS Version 15 Exit Routines	ERR	SC27-6787SC19-4208
IMS Version 15 Installation	INS	SC27-6788
IMS Version 15 Licensed Program Specifications	LPS	GC27-6799
IMS Version 15 Messages and Codes, Volume 1: DFS Messages	MC1	GC27-6789
IMS Version 15 Messages and Codes, Volume 2: Non-DFS Messages	MC2	GC27-6790
IMS Version 15 Messages and Codes, Volume 3: IMS Abend Codes	MC3	GC27-6791
IMS Version 15 Messages and Codes, Volume 4: IMS Component Codes	MC4	GC27-6792
IMS Version 15 Operations and Automation	OAG	SC27-6793
IMS Version 15 Release Planning	RPG	GC27-6794
IMS Version 15 System Administration	SAG	SC27-6795
IMS Version 15 System Definition	SDG	GC27-6796
IMS Version 15 System Programming APIs	SPR	SC27-6797
IMS Version 15 System Utilities	SUR	SC27-6798

© Copyright IBM Corp. 1974, 2020

Index

Special Characters

/ASSIGN command
    cold starting ISC sessions 493
    ETO limitations 98
/CHANGE command
    initiating an ISC session 493
/CLSDST command
    ISC TCP/IP sessions 604
/DEQUEUE command
    committing ISC output messages 508
/DISPLAY
    TMEMBER command 789
    TRANSACTION command 754
/DISPLAY command
    pending ISC output 494
    QSTOP state 422
/EXIT command
    ISC conversation mode errors 506
    terminating a conversation abnormally 395
/MSASSIGN command
    assigning physical link to logical link 669
/MSVERIFY command
    dynamic validation 704
    error responses 704
/OPNDST command
    ISC TCP/IP sessions 601
    relation to MODETBL keyword 418
/QUIESCE command
    ISC TCP/IP sessions 603
/SECURE OTMA NONE command 809
/SIGN command for ETO STSN devices 106
/SSR command 119
/START command
    making IMS ready 420
    starting a SLU P network 923
/START OTMA command 785
/STOP command 396
/STOP OTMA command 785

Numerics

119ABEND code 830
3270 copy command 440
3600 terminal
    supported as Finance or SLU P 899
3650 system, supported as SLU P 899
4700 terminals
    supported as Finance or SLU P 899
4730 Personal Banking Machine
    connecting in the XRF environment 905
8100 system, supported as SLU P 899

A

abend
    CICS 640
abend *(continued)*
    MSC conversations 689
access control environment element
    cached user ID aging value 861
accessibility
    features xxv
    keyboard shortcuts xxv
ACEE
    flood control 800
    IMS Connect
        specifying OTMA ACEE aging value 199
    OTMA, refreshing the ACEE 810
    refreshing OTMA ACEEs 810
ACK
    Send-only protocol 297
ACK option
    Finance Communication System 947
    SLU P 947
ACK timeout
    send-then-commit transactions (CM1) 803
acknowledgements
    OTMA ACK timeout intervals 801
administering
    Fast Path 784
    MSC 784
    queue control facility 785
    XRF 785
administration
    ETO 69
    MFS 430
    MSC (Multiple Systems Coupling) 685
advanced program-to-program communication (APPC)/IMS
    Error Extract Service 45
affinity
    definition 378
    in an MSC-IMSplex configuration
    715
affinity routing 732
aging value 861
allocating a session, CICS 620
alternate facility, CICS 625
alternate PCB, secondary transaction 389
alternate response PCB
    modifying 907
    use 907
ALTPCB
    IMS Connect and shared queues 174
    OTMA
        delivering output 789
        shared queues, retrieving output 789
    OTMA destination descriptor
        defining 776
        masking destination names 777
        role in IMS-to-IMS communications flow
        759
    shared queues and IMS Connect 174
ALTPCBE 789

API (application program interface)
explicit 369
implicit 369
API (application program interface), ISC
IMS use of VTAM 457
API (application program interface), ISC (intersystem communication)
asynchronous, CICS 589, 590, 623
synchronous, CICS 589, 590, 623
APPC
outbound LU
specifying 42
APPC (advanced program-to-program communication)
APPC/IMS 386
communications manager as the 30
destination code 387
destination structure 387
DFSAPPC 386
editing and formatting 425
IMS messages 386
input segment 387
LU 6.2 messages 386
queuing, SERIAL=YES option 386
segments, single or multiple 386
structure 387
switching 390
SYNCLVL=SYNCPT 30
types 386
APPC (advanced program-to-program communication)/IMS
API
explicit 35, 369
implicit 35, 369
APPC/MVS Administration utility example (ATBSDFMU) 41
application programs
CPI-C driven 35
modified IMS 35
remote standard IMS 37
standard IMS 35, 36
asynchronous output delivery 56
CPI Communications application program 25
CPI-C initialization 385
default conversation characteristics 56
DFSAPPC
message switching 54
option keywords 54
discardable messages 35
Error Extract Service 45
establishing APPC/IMS
APPC=Y or N 40
system definition 40
flood control 34
introduction 33
local service for remote APPC transaction 36, 38
LTERM interface 39
LU 6.2
devices 369
relationship to APPC/IMS 33
terminal support 385
LUADD option keywords 43
LUs, managing multiple 53
LUs, reassigning 53
MOD name 39
mode name 43
APPC (advanced program-to-program communication)/IMS *(continued)*
modified IMS application programs 37
MSC
application program failure and transaction recovery 50
intermediate IMS failure and transaction recovery 50
local IMS failure and transaction recovery 49
local transaction discardability 47
LU 6.2 recoverability flows 50
LU 6.2 sessions and transaction recovery 48
MSC links and transaction recovery 49
recovering transactions in APPC 47
remote IMS failure and transaction recovery 49
standard IMS applications 36
transaction point of failure 48
MSC (Multiple Systems Coupling)
processing remote transactions in an IMSplex 719
MSC and modified IMS applications 38
network-qualified LU name 52
nondiscardable messages 35
PARMLIB 43
partner LU name 43
qualifying LU names 52
RACF 57
recoverability 421
recoverable versus irrecoverable transactions 47
remote service for remote APPC transaction 38
restrictions 36
security 57
SIADD 43
side information
mode_name 43, 385
partner_LU_name 43, 385
TP_name 43, 385
sym_dest_name 43
Sync_Level options (NONE, CONFIRM, SYNCPT) 36, 37
SYS1.APPCSI 43
SYS1.APPCTP 40
SYS1.PARMLIB(APPCPMxx) example 43
timeout service 44
TP name 43
TP Profile
ATBSDFMU utility 40
definition 38
DFSTPPE0 42
DFSTPROF 41
dialog example 40
TSO ICQASRM0 41
VSAM data set 40
TPN (transaction program name) 42, 54, 55
transaction retry characteristics
deadlock 52
Fast Path retry conditions 52
lock reject 52
transaction security
ACCESS (EXECUTE) 57
UACC (NONE) 57
XRF 52
APPC/IMS
and protected transactions 836
APPC/MVS Administration utility example (ATBSDFMU) 41
application call processing 118
application program interface 457

application programming
OTMA
DL/I calls 816
application programs
callout function overview 856
CICS, with IMS 622
converting Finance to SLU P 901
correlating responses to asynchronous callout requests 857
definition 366
existing programs, with ISC 468
functions IMS provides 368
IMS
OTMA 754
ISC sample 657
remote 367
SLU P controller 901
XRF considerations, SLU P 901
application programs, z/OS
using the ODBA (Open Database Access) interface 747
using the Open Database Access (ODBA) interface 747
application threads 117
application-data section of OTMA message prefix 886
APPLID= parameter, ISC session definition 482
architected message 824
asynchronous callout requests
OTMA configuration overview 849
OTMA support overview 849
asynchronous messages
OTMA, rerouting output 846
OTMA, super member 847
asynchronous output
alternate client ID 335
ETO
valid destinations 102
implementing 326
managing 326
managing and controlling output messages 330
message flow 337
OTMA, managing output 845
OTMA, purging output 845
RESUME TPIPE, alternate client ID 335
retrieving output for another client 335
asynchronous output processing
commit mode 335
socket type 335
sync level 335
timer settings 335
asynchronous processing
ATTACH EB 461
IMS-CICS session 609
ISC execution mode 461
SCHEDULER FM header 544
ATCCONxx member (VTAM nodes) 419
ATCSTRyy member (VTAM start lists) 419
ATRABCK service 836
ATRACMT service 836
ATREINT service 836
ATTACH FM header
bit contents 557, 559
format 557, 565
MFS 550
parameter description 559, 565
process initiation 543
ATTACH FM header *(continued)*
with CICS 629
ATTACH FM headers
EB indicator 461
ATTIU parameter, FM header length 543
automated operator programs 381
autopaged output, synchronous 592

B

back-end subsystem
CICS 608
definition 447
IMS 605, 607
routing transactions 479
backout work unit, message resynchronization 491
balancing group
definition 380
routing codes and 380
balancing resource demand in MSC (Multiple Systems Coupling) 686
bandwidth
MSC links 686
basic edit
bypassing 436
editing options with ISC 435
IMS functions 434
IMS systems 425
input message segments 437
input messages 434
ISC messages 459
non-MFS programs 469
output message segments 437
output messages 435
SLU 1 transparent data 435
BB (begin bracket) indicator
definition 467
LUSTATUS command 528
use (figure) 905
begin bracket indicator
LUSTATUS command 528
use (figure) 905
BID command 515, 910, 918
BID option
caused by session termination 949
design considerations 910, 918
effects
display screen protection 914
MFS paging 914
output messages 911
bind
IMS-CICS session 637
ISC session
parallel 490
single 490
negotiable 490
rejected, ISC 536
requesting asynchronous process 544
BIND race 491
BINPDSB1= parameter, BINTRNDS option 435
BIS (bracket initiation stopped) command
IMS-CICS session 621
session shutdown 540
BPE (Base Primitive Environment)

BPE (Base Primitive Environment) *(continued)*
header data 237
header format 230
bracket and send/receive management
Finance Communication System
direction indicators 905
protocol 905
ISC, how determined 464
SLU P
direction indicators 905
bracket contention
invalid paging 917
resolving 491, 917
bracket initiation stopped (BIS) command
IMS-CICS session 621
session shutdown 540
bracket protocol
IMS 515, 932
input bracketing 936
input messages, ISC 515
output bracketing 942
output messages, ISC 518
bracket rejection
Finance Communication System 924, 949
ISC 537
SLU P 924, 949
BREAK code X'0811'
actions taken 949
buffer
MSC (Multiple Systems Coupling) considerations 685
MSC (Multiple Systems Coupling) linking 667
buffer sizes
BUFSIZE parameter 695
IMS-to-IMS sessions
479
ISC 484
MSC physical links 695
buffers
BUFSIZE parameter 695
MSC links
buffer size and format in bandwidth mode 708
buffer size and format in non-bandwidth mode 709
determining buffer size 707
format of link buffers 707
MSC links and bandwidth 686
MSC non-VTAM links 695
MSC VTAM links 695
sizes for MSC links 695
small buffer devices 438
BUFSIZE parameter for MSC LU 6.2 695

C

caching scheme, user ID 807
Callable Interface (C/I)
introduction to 889
calling out from IMS application programs
OTMA tpipe support for parallel RESUME TPIPE requests 769
tpipes, support for parallel RESUME TPIPE requests 769
callout
asynchronous
correlating a response 857
routing responses to IMS Connect client 858
callout *(continued)*
asynchronous *(continued)*
routing responses to input terminal 857
IBM MQ 856
IMS Connect support 201
OTMA
routing 854
OTMA tpipe support for parallel RESUME TPIPE requests 769
SOAP Gateway 855
synchronous
acknowledgment messages (ACK and NAK) for IMS Connect 205
acknowledgment messages, IMS Connect IRM_F3_REROUT 206
acknowledgment messages, IMS Connect NAK 206
acknowledgment messages, IMS Connect SYNCNAK 207
coding user-written IMS Connect clients 201
error responses, returning to IMS Connect 208
IMS Connect correlation token structure 239, 245
IMS Connect message format 203
IMS Connect, returning responses to 299
program switch 851
responses, returning to IMS Connect 299
RESUME TPIPE call 204
RESUME TPIPE error scenarios 205
retrieving callout requests 204
returning callout responses 207
send-only protocol 207
timeout, OTMA ACK 801
terminals
avoiding hung terminals 857
tpipes, support for parallel RESUME TPIPE requests 769
callout control data
format 246
callout function
asynchronous
correlating responses to callout requests 857
IMS Connect TCP/IP clients 854
IMS TM Resource Adapter
support overview 855
z/OS Sysplex Distributor
855
overview of application programs 856
callout messages
IMS Connect
RESUME TPIPE request 323
callout request 853
callout requests
OTMA
implementing asynchronous callout support 853
OTMA support 849
synchronous
MULTIRTP 858
OTMA configuration overview 849
OTMA support 850
parallel processing of RESUME TPIPE requests 858
callout support
asynchronous
programming IMS Connect clients 854
CANCEL command
paging errors 531, 538
protocol 523

CANCEL command *(continued)*
SLU P session 951
CANCEL request
sender ERP 536
candidate printers 440
capability exchange (CAPEX)
starting an ISC TCP/IP session from CICS 602
starting an ISC TCP/IP session with CICS 601
CAPEX (capability exchange)
starting an ISC TCP/IP session from CICS 602
starting an ISC TCP/IP session with CICS 601
CASCADE=
enabling IMS Connect cross-LPAR support for IMS TM transactions 355
CBresynch 840
CBresynch command 864
CCTL
connections for CICS 744
CD (change direction) indicator
definition 467
Finance Communication System 904
LUSTATUS command 528
send/receive protocol 905, 932
SLU P 904
soliciting 510
chained message communication sequence 936
chains, logical terminal 391
CHANGE (/CHANGE) command
initiating an ISC session 493
change direction (CD) indicator
Finance Communication System 904
send/receive protocol 905, 932
SLU P 904
channel-to-channel (CTC)
MSC (Multiple Systems Coupling) physical link type 667
MSC physical link
defining 696
CHASE command 524
CHECK security level 807
CICS
_CXX_LSYSLIB environment variable 20
accessing IMS databases 743
applications
running, with IMS Universal drivers 21
CCTL connections 744
configuration
DRA tasks 745
DFST2CIC DLL 20
DL/I support 744
DRA
coding PSBs 743
DRA configuration 745
IMS Universal drivers
running applications 21
IMS Universal drivers, type-2
configuring 19
IMS Universal JDBC driver, type-2
configuring 19
installing type-2 IMS Universal drivers 20
ISC
application-related concepts 622
functions supported 587
ISC (Intersystem Communication)
restarting TCP/IP sessions 604
CICS *(continued)*
ISC (Intersystem Communication) *(continued)*
session restart, TCP/IP 604
session termination, abnormal, TCP/IP 604
session termination, orderly, TCP/IP 603
session termination, TCP/IP 603
session termination, TCP/IP unconditional 604
TCP/IP session restart 604
TCP/IP session termination 603
TCP/IP session termination, abnormal 604
TCP/IP session termination, orderly 603
TCP/IP session termination, unconditional 604
terminating TCP/IP sessions 603
terminating TCP/IP sessions unconditionally 604
terminating TCP/IP sessions, orderly 603
ISC node
defining 613
ISC support
defining an ISC TCP/IP link to CICS 599
ISC TCP/IP
CICS front-end transaction types 604
ISC TCP/IP support
capability exchange (CAPEX) 601, 602
sessions, starting 601
sessions, starting from CICS 602
Makefile 20
sequential buffering 744
TCP/IP
ISC session restart 604
ISC session termination 603
ISC session termination, abnormal 604
ISC session termination, orderly 603
ISC session termination, unconditional 604
restarting ISC sessions 604
session restart, ISC 604
session termination, abnormal, ISC 604
session termination, ISC 603
session termination, ISC unconditional 604
terminating ISC sessions 603
terminating ISC sessions unconditionally 604
terminating ISC sessions, orderly 603
CICS resource definition 613
CICS-IMS communication
alternate facility 625
application coding for 622
asynchronous processing flow 609
ATTACH parameters 629
CICS transactions 620
coding function management headers 629
coding system definition options 613
Command Level API 589, 590
configuration 450
conversation mode 625
defining CICS transactions 620
device mapping function 450
facility
alternate 625
principal 625
functions
description 589, 590
overview 452
IMS commands 626
initiating sessions 620
integrity of session 637

CICS-IMS communication *(continued)*
LU 6.1 links
compatible nodes 614
description 613
Macro-Level Resource Definition 613
multiple links 618
Resource Definition Online 613
MFS support 634
mirror transaction 609
MSC links 451
passing data to IMS with ISC 453
preparing CICS tables 613
principal facility 625
processing flows
RECEIVE 608
RETRIEVE 610
SEND INVITE 605
SEND LAST 607
SEND/RECEIVE 605
START/RETRIEVE 610
recovery and restart 635
SCHEDULER parameters 632
session
binding 637
initiation 620
integrity 637
processing outstanding traffic 638
reestablishing 637
resynchronizing 637
sync points 627
termination 621
sync points 627
transactions
abnormal termination 623, 639
attributes supported 589, 590, 623
definition 620
CICS-ISC installation options 613
CIMS class 402
Class 1 terminals
overview 407
status recovery modes 399
XRF support 406
Class 2 terminals
overview 407
status recovery modes 400
XRF support 406
Class 3 terminals
overview 408
status recovery modes 400
XRF support 406
client
commands issued 838
high-performance access to IMS 756
number that can connect to OTMA 756
OTMA, in XRF environment 785
client communications
and IBM WebSphere 142
local option 142
client descriptors, OTMA
limit 778
Client_Bid resynch 840
client-bid request
message flow 832
message prefix contents 832
CLSDST (/CLSDST)
command
ISC TCP/IP sessions 604
CM0
OTMA, managing output 845
OTMA, purging output 845
OTMA, rerouting output 846
OTMA, super member 847
purge function 291
reroute function 293
CM1 (send-then-commit) transactions
ACK timeout 803
COBOL
XML conversion support
overview 154
coding CICS
applications for ISC 622
system definition macros 613
tables 613
coexistence
IMSplex and MSC 713
MSC and IMSplex 713
cold start, recovering ISC sessions 493
COMM macro statement
APPLID= keyword 416
RECANY= keyword 417
command
/DISPLAY TRANSACTION 754
CBresynch 840
issued by client 838
REPresynch 840
REQresynch 840
SRVresynch 840
TBresynch 840
Command Level API, CICS 589, 590
command recognition character (CRC) 119
commands
/CHANGE 135
/DBDUMP DATABASE 824
/DBRECOVERY AREA 824
/DISPLAY SUBSYS 135
/DISPLAY TMEMBER 789
/SECURE OTMA NONE 809
/SSR 119
/START AREA 824
/START REGION 824
/START SUBSYS 117
/STOP AREA 824
/STOP REGION 824
/STOP SUBSYS 117
asynchronous reply 592
CICS EXEC
asynchronous API 589, 590
creating DFC protocols 589, 590
functions available 589, 590
program flow 605
CICS-IMS session 626
CRC (command recognition character) 119
DEQUEUE TMEMBER 792
DISPLAY STATUS 792
DISPLAY TMEMBER 792
DISPLAY TRACE 792
IMS
parallel ISC sessions 460

commands *(continued)*
IMS *(continued)*
start system 420
issuing from an ISC session 460
LUSTATUS protocol, ISC 530
PRNs, not specified on IMS commands 562
recovery at ISC session failure 495
response requirements 947
securing with RACF 402
START TMEMBER 792
STOP TMEMBER 792
TRACE TMEMBER 792
transaction reply 605
used to start system 923
VTAM
BID 910
BIS 540, 621
CANCEL 523, 951
CHASE 524
commands and indicators 456, 902
LUSTATUS command 528, 950
LUSTATUS with CICS 592, 625
ready to receive (RTR) 531
Request-Recovery 951
RSHUT 531
RTR 950
SIGNAL 540, 951
commit
LUSTATUS command 530
mode 824
processing 824
sample flows 826
work unit 491, 636
Commit Mode
CM1 timeout value 283
Commit mode 0 283
Commit mode 1 283
commit mode 0
purge function 291
reroute function 293
commit-then-send
OTMA, managing output 845
OTMA, purging output 845
OTMA, rerouting output 846
OTMA, super member 847
purge function 291
reroute function 293
commit-then-send exchanges
OTMA message flow 834
OTMA message prefix contents 834
commit-then-send flow 826, 834
Common Queue Server (CQS)
checkpoint data set 375
client
exit routines 376
client, definition 373
definitions 373
overview 376
shared queue environment, in a 374
communications
IMS
prerequisite knowledge xxiii
communications controller 367, 368
communications network
communications network *(continued)*
components
responsibilities 366, 380
Finance Communication System 900
SLU P system 900
communications, establishing 372
COMPINOP state 396
component definition
LTERM naming 463, 907
parameter definitions 519
selection
input component 464, 907
output component 467, 907
SLU P input component 908
SLU P system 907
component protection
extended output 914
state 396
configurations
Finance Communication System 900
ISC 450
SLU P system 900
VTAM 419
connecting multiple IMS systems 685
connection
description 457, 904
IMS session parameters
ISC 488, 490
restriction against altering 904
XRF complex, establishing connections 905
connection factory
IMS Universal Database resource adapter 6
connections
IMS
prerequisite knowledge xxiii
contention, bracket 491, 917
context 817
control block
MFS 432
control block mapping
EEVT 123
control blocks
ISC 461
control character 438
control function, MFS 425
controlling output 439
Conversation Abnormal Termination exit routine 684
conversation mode
errors
ISC 505
explained 394
IMS-CICS 625
normal termination, ISC extension 506
restriction 479
conversational processing 389
conversational processing in MSC (Multiple Systems Coupling)
abend 689
planning for 687
remote destination verification 688
conversational programs
IMS Connect support 285
conversational transactions
OTMA, terminating in 806

conversational transactions *(continued)*
terminating in OTMA 806
conversations
OTMA and IMS 784
protected 817
conversion
COBOL, XML example 212
XML to COBOL example 212
coordinating performance information for MSC 711
COR token for synchronous callout 245
correlation
correlating responses to asynchronous callout requests 857
correlator token 759
coupling facility
definition 373
CPI Communications 23
CPI Communications application program
abnormal termination 26
APSB call 25
ATBCMTP verb 26
Backout call 26
backout processing 25
Commit call 25
commit processing 25
Db2 for z/OS plan name, use of 25
ESS Attach Facility 25
in-flight unit of recovery, definition 28
indoubt unit of recovery, definition 28
normal termination 25
programming requirements 27
pseudonym files 27
recovery 27
resolve-in-doubt processing 26
return codes 26
RTT 25
session failure 26
SQL calls 25
SRRCMIT 25
system restart 26
two-phase commit process and 28
CPI-C (Common Programming Interface for Communications)
APPC/IMS 385
initialization 385
side information 385
transactions 388
CQS (Common Queue Server)
checkpoint data set 375
client
exit routines 376
client, definition 373
definitions 373
overview 376
shared queue environment, in a 374
CRC (command recognition character) 119
CREATE MSNAME command
logical link paths 670
CREATE MSPLINK command
defining CTC physical links 696
defining MTM physical links 696
defining TCP/IP physical links 697
defining VTAM physical links 697
creating ESMT 121
creating work areas for ESAP 131
CRGGRM service 836
CRGSEIF service 836
CSM (complete status message)
format 243
CTC (channel-to-channel)
MSC (Multiple Systems Coupling) physical link type 667
MSC physical link
defining 696
CTXBEGC service 836
CTXEINT service 836
CTXSWCH service 836
customizing IMS 781

D

D descriptors, OTMA
limit 778
maximum 778
data descriptor FM header
IMS use 635
input format 565
output format 565
use on input 551
use on output 551
data flow
in an MSC (Multiple Systems Coupling) network 672
message switching 390
program-to-output terminal 389
program-to-program 389
terminal-to-program 388
data flow control (DFC) protocols
bracket and half-duplex 515
bracketing messages
input 515
chaining 524
commands
BID 515
ERP PURGE 525
error handling 505
examples 643
exception response 505
half-session synchronization 506
input messages, backing out 507
paged message errors 531
PURGE after exception response 525
recoverability aided by LWA 506
response requirements 506
sense codes 538
symmetrical session shutdown 540
sync point and response
CICS-IMS 627
input 510
output 512
data flow reset, handling errors 536
data partitioning, MSC (multiple system coupling) 667
data stream profile, ATTACH FM header 560
data transmission
definite responses, ISC 455
error handling 948
error recovery procedure 635
exception responses, protocol
CICS-IMS 626, 635
Finance Communication System 902

data transmission *(continued)*
exception responses, protocol *(continued)*
ISC 455
messages
input 932
output 936
response requirements
IMS commands and indicators 456, 902
IMS message switches 510, 947
irrecoverable-inquiry transactions 510, 946
LU 6.2 application program 421
MFS control requests 947
recoverable-inquiry transactions 510, 946
summary 944
VTAM commands and indicators 456, 902
SLU P, message data types 931
data transparency 459
data types, IMS advanced function network 931
data-set-full condition 440
database division 686
database recovery adapter (DRA)
setup 747
database recovery adapter (DRA) start-up table
using with ODBA (Open Database Access) 747
using with Open Database Access (ODBA) 747
Database Resource Adapter (DRA)
connections
configuring 741
DB2 Attach Facility
preparing the system 111
Db2 for z/OS
CPI-C
plan name, use of 25
Java dependent regions
accessing Db2 for z/OS
111
DBCTL
initializing 745
OTMA unsupported 792
DBFHAGU0 (Input Edit/Routing exit routine) 381
DCCTL (Data Communication Control) 371
DCCTL (Data Communications Control)
generation 371
IMS BTS 371
procedures 371
TM batch 371
dead letter queue HWS$DLQ 342
DEADQ STATUS 98
deblocking algorithm, ATTACH FM header 560
deferred program switch 390
defining
IMS 480
ISC node 480
definite responses
Finance Communication System 902
response requirements (figure) 944
SLU P 902
demand-paged messages
controlling
QMODEL FM headers 552
dependent region connections 137
dependent regions
commit-then-send transactions 794
OTMA usage 794
dependent regions *(continued)*
send-then-commit transactions 794
DEQUEUE (/DEQUEUE) command
committing ISC output messages 508
DEQUEUE command
FORCESS versus SYNCSESS 494
IMS-to-IMS ISC session 494
dequeuing messages, implications 493
descriptors
DFSOTMA descriptor 777
logon
ETO 63
OTMA
limit 778
maximum 778
OTMA client descriptor 773
OTMA destination descriptor
defining 776
masking destination names 777
OTMA, specifying 773
destination
determination message 782
destination descriptors, OTMA
limit 778
destination process name (DPN)
ATTACH FM header 561
message routing, ISC 469
SCHEDULER FM header 561
destination queue name, ATTACH FM header 564
Destination Resolution exit routine (DFSYDRU0) 781
destination system (MSC) 676
device class control 438
device input format (DIF) 432
device output format (DOF) 432
DFC (data flow control) protocols
bracket and half-duplex 515
bracketing messages
input 515
output 518
chaining 524
commands
BID 515
BIS 540
CANCEL 523
CHASE 524
LUSTATUS 528
RSHUT 531
RTR 531
SBI 540
SIGNAL 540
ERP PURGE 525
error handling
conversation mode 506
response mode 505
selective receiver ERP 531
examples 643
exception response 505
half-session synchronization 506
input messages, backing out 507
paged message errors 531
PURGE after exception response 525
recoverability aided by LWA 506
response requirements
irrecoverable-inquiry transactions 506

DFC (data flow control) protocols *(continued)*
response requirements *(continued)*
recoverable-inquiry transactions 506
sense codes 538
symmetrical session shutdown 540
sync point and response
availability 507
CICS-IMS 627
exceptions for synchronous input 507
input 510
irrecoverable messages 510
MFS output messages, ISC 508
output 512
recoverable messages 508
requested on input 507
requested on output 508
DFS3650I (session status message) 417
DFSAERA0 module 748
DFSAERG0 module 748
DFSAERM0 module 748
DFSAPPC message switching 54
DFSCCMD0 781
DFSCDLI0 module 748
DFSCMUX0 exit routine 702
DFSCTRN0 781
DFSEMODL macro 121
DFSEWAL macro 124
DFSFEBJ0 (Front-End Switch exit routine)
special support 447
DFSMSCE0 781
DFSOTMA descriptor 777
DFSPBxxx
OTMA parameters
GRNAME 768
OTMA= 768
OTMAASY 772
OTMAMD 772
OTMANM 768
OTMASE 769
OTMASP 772
DFSQSP0 781
DFSSIML0 exit routine 441
DFSYDRU0 exit routine 781
DFSYIOE0 exit routine 781
DFSYMSG DSECT 859
DFSYPRX0 exit routine 781
DFSYRTUX exit routine 781
DIF (device input format) 432
direct-control subsystem 622
directed routing
MSC 683
password not passed across link 705
disabling enforcement
resource name uniqueness 379
resource type consistency 380
DISPLAY (/DISPLAY) command
QSTOP state 422
display screen protection
BID option 914
definition 914
Finance Communication System terminals 914
MFS 914, 915
NOBID option 914
Distributed Presentation Management (DPM)
Distributed Presentation Management (DPM) *(continued)*
FM headers 548
option 437
OPTIONS=DPAGE or PPAGE 548, 939
output 939
SLU P supports 911
Distributed Relational Database Architecture (DRDA)
IMS Connect support 145
distributed security credential propagation
IMS Connect 179
distributed transaction processing, ISC 447, 469
distributed two-phase commit 346
DL/I
OTMA
calls used 816
documentation
exit routines 701
terminal profiles 415
DOF (device output format) 432
DPM (Distributed Presentation Management)
FM headers 548
option 437
OPTIONS=DPAGE or PPAGE 548, 939
output 939
SLU P supports 911
DPN (destination process name)
ATTACH FM header 561
message routing, ISC 469
SCHEDULER FM header 561
DR2 response
exception
Fast Path 920
irrecoverable-inquiry transactions 944
MFS output 944
nonrecoverable output 908
when MOD does not specify PAGE=YES 929, 945
requirements
MFS paged output messages, not Fast Path 908
normal IMS output messages 908
recoverable-inquiry transaction 946
DRA
CICS
DRA configuration 745
DRA (database recovery adapter)
setup 747
DRA (database recovery adapter) start-up table
using with ODBA (Open Database Access) 747
using with Open Database Access (ODBA) 747
DRA (Database Resource Adapter)
connections
configuring 741
DRD
enabling for MSC 690, 691
DRDA (Distributed Relational Database Architecture)
IMS Connect support 145
DSCA operand, DEV statement 441
duplicate databases 686
dynamic allocation VTAM subpools 463
dynamic terminals 369

E

EB (end bracket) indicator
definition 467

EB (end bracket) indicator *(continued)*
LUSTATUS command 528
use (figure) 905
used with ATTACH 461
ECSA
OTMA usage 793
edit
basic edit during ISC 459
ISC
default editor 459
MFS
CICS 592
during ISC 459
options
data communication exit routine 459
input and output 459
ISC input and output, list 448
editing facilities
invoking FM headers 542
ISC overview 459
editing messages
basic edit and nongraphic messages 437
bypassing Basic or MFS editing 436
editing performed by IMS 434
output segments 437
transparency option 436
editing options
COMPTn parameter of TERMINAL macro 484
OUTBUF parameter of TERMINAL macro 484, 916
SLU P, COMPTn parameter of TERMINAL macro 916
types
DPM-An 916,
939
MFS-SCS1 916
SCS1 916
EEVPEEA 129
EEVPEWA 131
EEVT (external entry vector table) 130
EEVT mapping 130
EMH (expedited message handler)
queue option
overview 380
EMH buffer 404
EMHL control region initialization parameter 404
encryption
OTMA restriction 792
end bracket indicator
definition 467
LUSTATUS command 528
use (figure) 905
used with ATTACH 461
environments supported, IMS 754
ERP (error recovery procedure)
CICS-IMS session 635
extended 635
FM header
format 566
function management header 635
implemented by IMS 902
selective receiver
sense codes 532
ERP PURGE, after ISC exception response 525
Error Extract Service
APPC/IMS 45
error handling
BREAK code X'0811'
output messages 949
CANCEL command
use 951
when sent by IMS 951
controller-detected errors
system sense field 948
use 949
user sense field 950
error messages 548
FM header 543
IMS-detected errors 948
IMS-issued error messages in XRF complex 948
ISC
CICS-IMS session 635
paging errors, during data flow reset state 536
paging errors, for nonpaged messages 538
paging errors, for paged messages 531
selective receiver ERP 531
sender ERP 536
sender ERP sense codes 537
length of message 543
LUSTATUS command 529, 950
MFS-detected errors 548
queuing messages 543
RQR command 951
SIGNAL command 951
SLU P
IMS-detected errors 948
VTAM logical unit status command 625
error messages
conditions causing
BB-only specified on input 936
BB-only specified on output 942
length 543, 948
MFS-detected errors 548
error recovery procedure (ERP)
CICS-IMS session 635
extended 635
FM header
format 566
function management header 635
implemented by IMS 902
selective receiver
sense codes 532
errors
ISC session termination, causing 536
ESMT (external subsystem module table)
creating 121
loading external subsystem modules 130
work area definitions 124
establishing connection 457
establishing connections for external subsystems 116
ETO
device characteristics table
building 87
IDC0 Trace facility 104
recovering ETO terminals with XRF 84
terminals
planning for growth 73
ETO (Extended Terminal Option)
/SIGN command for ETO STSN devices 106
3275 devices 75

ETO (Extended Terminal Option) *(continued)*
3600/Finance 105
ABENDU0015 89
advantages
availability 69
LTERMs 69
algorithm
logon descriptor 83
LTERM allocation 94
associated printing techniques 95
asynchronous output
destinations, valid 102
autologoff 99
autologon 101
autosignoff 99
benefits of using 61
commands that reset status and release control blocks 94
commands that retain status 94
common logon descriptors 81
concepts
summary 66
conversations 106, 390
customizing 66, 70
dead-letter queue 101, 102
default CINIT/BIND user data formats 90
defining
autosignoff and autologoff timer 101
parameters 98
deleting control blocks
after logoff 103
after signoff 103
delivering output to non-originating terminal 102
descriptors
added 89
creating during system definition 81
definition 63
deleted 89
introduction to coding 80
logon 81
logon, definition 63
MFS 86
MSC (Multiple Systems Coupling) 88
updated 89
user 84
using 66
VTAM TERMINAL macro 81
device characteristics table 75, 86
device type, defining 74
DFS2085 91
DFS3641W 89
DFS3645 91
DFS3649A 92
DFS3650I 93
DFS3672 91
DFSINSX0 86
DFSSGNX0 86
DFSUSER descriptor 86
DLQT 102
dynamic terminal
definition 61
dynamic terminals
static terminals, using together 72
dynamic user, definition 62
ETO (Extended Terminal Option) *(continued)*
exit routines
Signon exit routine (DFSSGNX0) 96
using 66
exit routines, coding 88
exit routines, list of 88
guideline selection
logon descriptors 82
LOGOND parameter 82
initialization
descriptor validation 89
DFSINTX0 89
ISC TCP/IP 61
logging off 103
logon
/OPNDST command 89
dynamic, limiting to specific terminal types 90
INITOTHER 89
INITSELF 89
signon data 89
USS LOGON 89
logon descriptors
creating during system definition 82
NTO, 3600/Finance terminals 84
LTERM
creating and reusing control blocks 90
LTERM with specific destination 93
LU 2
devices 75
screen size and model information 75
LU 6.1 (ISC) terminals 104
MFS 79
MFS device characteristics table 75, 86
MFS device descriptors
MFSDCT utility (DFSUTB00) 87
MFSDCT utility 75
MODETBL on ETO logon descriptor 418
MSC (Multiple Systems Coupling)
descriptor 80
MSNAME macro 80
support 80
multiple signons 91
node user descriptor 85
non-SNA 3270 devices
printers and displays 74
screen size and model information 75
NTO devices 75
output
assigning 101
inadvertent output data streams 103
overview 61
planning
LU2 76
operations 80
user IDs 79
user queue names 79
printers
defining 96
direct printing 95
overview 95
printer node names 96
sharing printers 97
queue (dynamic user message), definition 61
RACF 370

ETO (Extended Terminal Option) *(continued)*
recommendations 86
requirements 70
response mode 106
restrictions 70
screen definitions examples (non-SNA 3270)
display (model specified) 77, 78
display (screen size specified) 78
LU0 video 77
model 2 printer 77
screen size control byte 75
security
static versus dynamic terminals 78
shared printing 97
signing off, definition 103
signing on, definition 91
signon
LTERM allocation 91
providing signon data 91
Signon exit routine (DFSSGNX0) 96
SLU P 105
SNA commands 89
special processing modes 94
starting ETO 89
static terminal, definition 61
static terminals
dynamic terminals, using together 72
storing descriptors 81
structure
terminal, definition 62
user, definition 62
structure, creation and deletion 65
STSN terminals
overview 105
support for /SIGN command 106
system definition 81
terminal
definition 61
terminal-LTERM relationship 93
terminals
using static and dynamic terminals together 72
terminology 61
undeliverable data, dead-letter queue
/DISPLAY STATUS USER command 102
/DISPLAY USER DEADQ command 102
user descriptors
creating during system definition 84
criteria for selection 84
DFSUSER 86
installation created 85
installation-created 84
node user descriptor 84, 85
VTAM CINIT LUNAME 82
VTAM considerations
logon CINIT session control blocks 74
VTAM PSERVIC parameters 74
EWAL 124
exception response
DFC command, purging 505
ISC response mode errors 505
protocol
response requirements (figure) 944
exception response *(continued)*
protocol *(continued)*
VTAM facilities 455
exclusive mode 395
EXEC parameter, SSM 114
execution mode
recovery at ISC session failure 495
specification, ISC 461, 462
supported by ISC, list 448
execution modes
CICS communication 589, 590
ISC 589, 590
EXIT (/EXIT) command
ISC conversation mode errors 506
terminating a conversation abnormally 395
exit interface block (XIB) 163
exit interface block (XIB1) 167
exit interface block ODBM data store entries 168
exit interface blocks
IMS Connect
XIBDS 164
exit routines
affinity routing 732
for ETO 88
for MSC 701
IMS Connect
overview 159
overview, function-specific exit routines 162
overview, user message exit routines 160
IMSplex affinity routing 732
IMSplex message routing 730
MSC message routing 730
routing 730
TM and MSC Message Routing and Control user exit routine 730
exit routines, with OTMA 781
expedited message handler (EMH)
queue option
overview 380
explanation of stopped status 137
express_context_interest service. 836
extended output component protection 914
Extended Recovery Facility 785
Extended Recovery Facility (XRF)
Class 1 terminals 409
Class 2 terminals 407
Class 3 terminals 408
class of service 408
classes of terminals 406
definition 405
establishing communication
Finance Communication System 905
ISC 489
SLU P 905
system takeover considerations 905
master terminals 393
MNPS 406
recovery modes 399
SLU P application program 901
takeover considerations
Finance Communication System 905
SLU P 905
terminal support 385, 408
terminals in 406

Extended Terminal Option (ETO)
/SIGN command for ETO STSN devices 106
3275 devices 75
3600/Finance 105
ABENDU0015 89
advantages
availability 69
LTERMs 69
algorithm
logon descriptor 83
LTERM allocation 94
associated printing techniques 95
asynchronous output
destinations, valid 102
autologoff 99
autologon 101
autosignoff 99
benefits of using 61
coding descriptors
introduction 80
commands that reset status and release control blocks 94
commands that retain status 94
common logon descriptors 81
concepts
summary 66
conversations 106, 390
creating descriptors 81
customizing 66, 70
dead-letter queue 101, 102
default CINIT/BIND user data formats 90
defining
autosignoff and autologoff timer 101
parameters 98
deleting control blocks
after logoff 103
after signoff 103
delivering output to non-originating terminal 102
descriptors
added 89
definition 63
deleted 89
logon 81
logon, definition 63
MFS 86
MSC (Multiple Systems Coupling) 88
updated 89
user 84
using 66
VTAM TERMINAL macro 81
device characteristics table 75, 86
device type, defining 74
DFS2085 91
DFS3641W 89
DFS3645 91
DFS3649A 92
DFS3650I 93
DFS3672 91
DFSINSX0 86
DFSSGNX0 86
DFSUSER descriptor 86
DLQT 102
dynamic terminal
definition 61
Extended Terminal Option (ETO) *(continued)*
dynamic terminals
static terminals, using together 72
dynamic user, definition 62
exit routines
Signon exit routine (DFSSGNX0) 96
using 66
exit routines, coding 88
exit routines, list of 88
guideline selection
logon descriptors 82
LOGOND parameter 82
initialization
descriptor validation 89
DFSINTX0 89
ISC TCP/IP 61
logging off 103
logon
dynamic, limiting to specific terminal types 90
logon descriptors
creating during system definition 82
NTO, 3600/Finance terminals 84
LTERM
creating and reusing control blocks 90
LTERM with specific destination 93
LU 2
devices 75
screen size and model information 75
LU 6.1 (ISC) terminals 104
MFS 79
MFS device characteristics table
screen size and model information 75
MFS device descriptors
MFSDCT utility (DFSUTB00) 87
MFSDCT utility 75
MODETBL on ETO logon descriptor 418
MSC (Multiple Systems Coupling)
descriptor 80
MSNAME macro 80
support 80
multiple signons 91
node user descriptor 85
non-SNA 3270 devices
printers and displays 74
screen size and model information 75
NTO devices 75
output
assigning 101
inadvertent output data streams 103
overview 61
planning
LU2 76
operations 80
user IDs 79
user queue names 79
printers
defining 96
direct printing 95
overview 95
printer node names 96
sharing printers 97
queue (dynamic user message), definition 61
RACF 370
recommendations 86

Extended Terminal Option (ETO) *(continued)*
reduction of time of system definition 370
requirements 70
response mode 106
restrictions 70
screen definitions examples (non-SNA 3270) 77
screen size control byte 75
security
signon 78
shared printing 97
signing off, definition 103
signing on, definition 91
signon
providing signon data 91
Signon exit routine (DFSSGNX0) 96
SLU P 105
SNA commands 89
special processing modes 94
starting ETO 89
static terminal, definition 61
static terminals
dynamic terminals, using together 72
storing descriptors 81
structure
terminal, definition 62
user, definition 62
structure, creation and deletion 65
STSN terminals
overview 105
support for /SIGN command
106
system definition 81
terminal
definition 61
terminal-LTERM relationship 93
terminals
using static and dynamic terminals together 72
terminology 61
undeliverable data, dead-letter queue 102
user descriptors
creating during system definition 84
criteria for selection 84
DFSUSER 86
installation created 85
installation-created 84
node user descriptor 84, 85
VTAM CINIT LUNAME 82
VTAM considerations
logon CINIT session control blocks 74
VTAM PSERVIC parameters 74
external entry vector table (EEVT) 130
external execution mode, ISC 461
external subsystem attach facility
accessing multiple external subsystems 135
application call processing 118
application threads 117
attach processing
overview 115
CHANGE command 135
commands
/START SUBSYS
136
/STOP SUBSYS 136
connection initiation 131
external subsystem attach facility *(continued)*
connections
dependent regions 132
CRC 119
creating ESMT 121
creating work areas for ESAP 131
deferring control region identify 132
dependent region connections 137
dependent regions
establishing connections 132
DFSEMODL macro 121
DFSEWAL macro 124
DISPLAY SUBSYS command 135
EEVPEEA 129
EEVPEWA 131
EEVT 130
EEVT mapping 130
ESMT
creating 121
description 114
establishing connections 116
EWAL 124
exit routine interface 114
explanation of stopped status 137
external subsystem command support 119
functions supplied by external subsystem 114
identify process 116
IMS services available to ESAP 119
INQ parameter 118
language interface module 134
loading ESAP 129
loading external subsystem modules 130
notify message 116, 133
OASN 135
overview 113
PDSE 121
processing 129
recovery coordinator 113
recovery token 118, 135
resource coordination 118
RTT 114
signon process 116
specifying external subsystems to IMS 114
SSR command 136
subsystem connections 116
subsystem termination 136
terminating connections 117
termination ECB 117, 136
termination requested by external subsystem 136
thread 116
token 113
two phase commit process 118
unique language interface entry points 134
user authorization processing 116
External Subsystem Attachment Package (ESAP) 114
external subsystem command support 119
external subsystem module table
PDSE 121
external subsystem, definition 370
external subsystems
attach facilities 109
callout request 853
connecting to 109
extracting multiple system transaction statistics 712

F

facilities
  available on IMS-CICS session 589, 590, 623
  component definition 907, 908
  display screen protection 914
  Finance Communication System 907
  ISC 459
  Message Format Service 911
  message recovery 913
  SLU P 907
  terminal-response mode 908
  test mode for ISC 460
failure
  CICS-IMS session 636
  ISC, recovering from in-bracket 494
  system, during CICS-IMS session 636
  while in-brackets 492
Fast Path
  EMH buffer size 404
  Input Edit/Routing exit routine (DBFHAGU0) 381
  input message 908
  messages 369
  MFS NEXTMSG or NEXTMSGP control commands 908
  MFS paged output messages 908
  NFPACK option on Terminal macro 920, 921
  output messages 908
  overview 369
  parameters
    FORCERESP 919
    FPACK/NFPACK 919
    OPTACK 919
  recoverable-inquiry transactions 421, 945
  recovery 399
  restrictions for applications 381
  routing code 380
  RTR command 920
  TERMINAL macro
    options 919
    required parameters 919
  terminal-response mode 908
  terminals 404
Fast Path EMH
  shared queues environment
    overview 380
Fast Path, administering 784
Finance communication 897
Finance Communication System
  application program
    converting Finance to SLU P 901
    functions available 900
    MFS considerations 900
  bracket and send/receive protocols 905, 932
  facilities 907
  major parts 899
  message resynchronization
    controller application program 924
  sample configuration 900
  system takeover considerations 905
  terminals supported 899
  VTAM
    commands and indicators 902
    facilities 902
Finance Communication System *(continued)*
  workstations 900
  XRF complex, establishing connections 905
first speaker
  ISC contention winner 458
  secondary logical units 458
flood control
  ACEE 800
  OTMA 800
flow
  commit-then-send 826, 834
  of data
    in an MSC (Multiple Systems Coupling) network 672
  of resynchronization 841
  send-then-commit 829
  send-then-commit with Confirm 830
FM (function management) header
  error recovery procedure (ERP)
    format 566
FM (function management) headers
  coding for use with CICS 629
  Finance Communication System 933, 939
  IMS, data descriptor 635
  input FM header length
    ATTACH FM header 543
  input message 459
  input process names 563
  inserted by IMS 437
  ISC
    ATTACH 543
    ATTACH, MFS 550
    data descriptor 551, 565
    DPM messages, MFS 548
    error recovery procedure 543, 635
    initiating a process 543
    Input QMODEL FM Headers 553
    introduction 541
    invoking ISC edit 542
    MFS 547
    QMODEL 552, 567
    QMODEL, CICS 634
    RAP 544
    RAP, MFS 555
    Reply QMODEL FM Headers 554
    SCHEDULER 544, 632
    SCHEDULER, MFS 550
    synchronous processing 543
    SYSMSG 545
    system message 545, 635
  message descriptor byte format
    input 933, 935
    output 939, 941
  MFS 634
  output FM header 939
  output message 461
  output process names 563
  processing mode requested 461
  security 449
  SLU P 934, 940, 942
  type X'42' (SLU P) 935
FM headers 437
FMH 437
format, message
  control characters in message prefix 438

format, message *(continued)*
creating with SDF II 432
transparency option 436
forms 439
FPACK/NFPACK option, Fast Path 919
front-end subsystem
CICS
SEND LAST command 607
SEND/RECEIVE command 605
START/RETRIEVE command 610
function of 447
IMS 608, 612
Front-End Switch
OTMA, unsupported 792
Front-End Switch exit routine (DFSFEBJ0)
special support 447
FULL security level 807
full-duplex message flow 759
function abort
detecting loop, ISC 537
indicated on LUSTATUS command 529
function management (FM) headers
coding for use with CICS 629
error recovery procedure (ERP)
format 566
Finance Communication System 933, 939
IMS, data descriptor 635
input FM header length
ATTACH FM headers 543
input message 459
input process names 563
inserted by IMS 437
ISC
ATTACH 543
ATTACH, MFS 550
data descriptor 551, 565
DPM messages, MFS 548
error recovery procedure 543, 635
format reference 557
header format 557
initiating a process 543
Input QMODEL FM Headers 553
introduction 541
invoking ISC edit 542
MFS 547
QMODEL 552, 567
QMODEL, CICS 634
RAP 544
RAP, MFS 555
Reply QMODEL FM Headers 554
SCHEDULER 544, 632
SCHEDULER, MFS 550
supported by IMS, summary 541
synchronous processing 543
SYSMSG 545
system message 545, 635
message descriptor byte format
input 933, 935
output 939, 941
MFS 634
output message 461
output process names 563
processing mode requested 461
security 449
function management (FM) headers *(continued)*
SLU P 934, 940, 942
type X'42' (SLU P) 935
functions
ISC and CICS 452, 589, 590
ISC versus MSC 446

G

generic resource groups
benefits 377
definition 378
overview 377
generic resource member, definition 378
generic resource name
definition 378
generic resources
MSC TCP/IP 726
TCP/IP
affinity management for MSC 726
affinity persistence for MSC 728
affinity, clearing for MSC 728
affinity, clearing in IMS Connect 729
IMS Connect support 152
MSC affinity management 726
MSC affinity persistence 728
MSC affinity, clearing 728
MSC affinity, clearing in IMS Connect 729
MSC and XRF 729
XRF and MSC 729
GRAPHIC= parameter
NO option 437
SEG statement 437
GRNAME parameter 768

H

half session
definition 487, 490
pairs 485, 490
synchronization, ISC 506
half-duplex message flow 759
half-duplex protocol, IMS use 515
HANDLE CONDITION, CICS 621
headers
input message 437
ISC 437
output message 437
type X'42'
component identification 935
data bytes 942
message descriptor byte 935, 941
HIOP (high input/output pool)
VTAM output buffers
RECASZ execution parameter 417
HIOP storage pool
temporary shortage, reregistering OTMA 787
hold queue, asynchronous
retrieving asynchronous output for alternate client IDs 335
horizontal partitioning in MSC (Multiple Systems Coupling) 667
HWS$DLQ 342

HWSIMSCB macro 213
HWSJAVA0
  local option client communication 142
HWSOMCTL DSECT 251
HWSOMUSR DSECT 274
HWSSMPL0
  and IRM 218
  message structures in simple flow 248
HWSSMPL1
  and IRM 218
  message structures in simple flow 248
HWSXIB macro 163
HWSXIB1 macro 167
HWSXIBDS macro 164
HWSXIBOD macro 168

I

IBM MQ
  OTMA
    synchronized tpipes, defining 772
ICAL
  synchronous program switch 851
identify process 116
immediate program switch 390
immediate session termination 499
IMS
  control region size and OTMA 793
  conversation
    and commit-then-send mode 825
  device support with OTMA 756
  DFSnnnnn messages 341
  high-performance access 756
  IMS.ADFSMAC 859
  OTMAASY start-up parameter 819
  processing protected transactions 836
  recoverable transactions 295
  scheduler message block (SMB)
    OTMA 757
  standard flow 826
  transactions
    using a nonsynchronized tpipe 838
    using a synchronized tpipe 838
  z/OS Resource Recovery Services exits supported 836
IMS connect
  BPE header format 230
  header format 229
IMS Connect
  affinity
    clearing, MSC TCP/IP links 729
  alternate client ID 335
  ALTPCB output and shared queues 174
  asynchronous callout requests from TCP/IP clients 854
  asynchronous callout support
    programming 854
  asynchronous OTMA output
    retrieving 322
  asynchronous output
    auto message control 333
    grouping 336
    noauto message control 332
    nooption message control 332
    requesting from end-user application 327
IMS Connect *(continued)*
  asynchronous output *(continued)*
    RESUME TPIPE request 323
    sharing 336
    single message control 331
    single with wait 331
  asynchronous output support 335
  availability, verifying 342
  calling out to external services from IMS 853
  callout
    coding user-written clients for synchronous callout 201
    RESUME TPIPE call for synchronous callout 204
    RESUME TPIPE error scenarios 205
    retrieving synchronous callout requests 204
    returning synchronous callout responses 207
    send-only protocol 207
    synchronous, message format 203
  callout requests
    retrieving from OTMA 322
  callout support 201
  callout support, synchronous
    acknowledgment messages (ACK and NAK) 205
    acknowledgment messages, IRM_F3_REROUT 206
    acknowledgment messages, NAK 206
    acknowledgment messages, SYNCNAK 207
    returning error responses 208
  cancel timer 319
  changing RACF password phrases 191
  changing RACF passwords 190
  client call flows 338
  client communications, restrictions for local 142
  client communications, TCP/IP 142
  client IDs
    avoiding duplicates 307
    canceling 307
  CM0
    ignore DL/I PURG call 232
  CM0, purge function 291
  CM0, reroute function 293
  CM1 ACK/NAK timeout value 283
  commit mode 0, purge function 291
  commit mode 0, reroute function 293
  commit-then-send
    ignore DL/I PURG call 232
  commit-then-send, purge function 291
  commit-then-send, reroute function 293
  conversational program support 285
  conversational protocols
    send-then-commit, sync level=confirm, ACK response 289
    send-then-commit, sync level=confirm, NAK response 290
    send-then-commit, sync level=none, client terminated transaction 288
    send-then-commit, sync level=none, program terminated transaction 287
  data stores
    status 164
  data stores, IMS DB
    status 168
  dead letter queue 342
  defining
    overview 157

IMS Connect *(continued)*
DFS messages
client response 340
Distributed Relational Database Architecture (DRDA) 145
exit interface block
format, IMS DB connections 168
XIB1 format 168
exit interface block (XIB)
format of entry 164
exit interface block (XIB1) 167
exit interface block data store entry
format of entry 165
exit interface block ODBM data store (XIBOD) 168
exit interface blocks
format of XIBOD 169
XIBDS 164
exit routines
exit interface block (XIB) 163
exit interface block (XIB1) 167
input message format on return from exit 235
macro support 162
overview 159
overview, user message exit routines 160
exit routines, function specific
overview 162
exit routines, security 183
HWS$DLQ 342
HWSJAVA0
user-defined messages 161
HWSOMCTL DSECT 251
HWSSMPL0
message structures in simple flow 248
HWSSMPL1
message structures in simple flow 248
HWSSOAP1 154
HWSXIB macro 163
HWSXIB1 macro 167
HWSXIBDS macro 164
HWSXIBOD macro 168
IBM MQ 856
ignore PURG call for CM0 multi-segment output 232
IMS TM Resource Adapter
callout support 855
duplicate client IDs 146
generated client IDs 146
message structures 233
support overview 146
two-phase commit support 352
IMS Universal drivers
alias name 146
connection routing 146
one-phase commit 349
two-phase commit support 346
IMS-to-IMS TCP/IP communications
introduction 150
MSC support 151
OTMA support 153
overview 150
reconnecting automatically 154
socket connections 302
super member support 153
TCP/IP generic resource support 152
termination scenarios 302
IMS Connect *(continued)*
IMSplex support 173
intervals for message timer 312
introduction 141
invoking
overview 157
IRM
fixed portion 214
IRM_F3_IPURG 232
ISC
overview of support 148
ISC support
defining an ISC TCP/IP link 598
local option 141
macros 162
MAXSOC parameter
RESVSOC impact 306
usage 304
message formats
IRM extensions 227
IRM fixed portion 214
output to message exit 238
message structures
examples 248
message timeout intervals 311
message timer
canceling 319
mixed-case passwords 192
MSC
affinity, clearing 729
ODBM
status 168
OMHDRIPG 232
Operations Manager
command requests 147
support overview 147
OTMA conversational protocol 285
OTMA conversational protocols
send-then-commit, sync level=none 285, 286
OTMA destination descriptor 774
OTMA IMS-to-IMS TCP/IP message flow 759
OTMA message header
HWSOMCTL DSECT 251
message control fields 251
notes 281
security data fields 270
state data fields 257
OTMA message headers
format of user data section 274
OTMA timeout for CM0 acknowledgments 319
output message structure
to clients 232
overview 141
passing distributed security credential 179
passing network security credentials 179
password management 190
ping function 342
PING response 244
protocol level 284
protocols
parallel RESUME TPIPE requests implementing 329
parallel RESUME TPIPE requests, diagnosing problems 329
RESUME TPIPE request 323

IMS Connect *(continued)*
protocols *(continued)*
RESUME TPIPE, parallel processing 327
RESUME TPIPE, parallel processing, enabling 328
send only 296
send only for callout responses 299
purge function 291
purging multiple-message output 292
purging output 291
purging undeliverable output
HWSSMPL0 and HWSSMPL1 292
when output is purged 292
RACF support 175
RACF user ID cache 178
RACF, default user ID 178
RACF, enabling direct support 176
RACF, enabling statistics 177
RACF, generic return code or message 176
reconnecting automatically
IMS-to-IMS TCP/IP communications 154
reroute function 293
rerouting commit-then-send output
purging multiple-message output 295
specifying a destination 294
when output is routed 294
rerouting output 293
RESUME TPIPE
synchronous callout, coding for 204
RESUME TPIPE call
and timeout 334
RESUME TPIPE protocol
example flows 324
RESUME TPIPE request 322, 323
RESUME TPIPE requests
parallel processing 327
parallel processing, diagnosing problems 329
parallel processing, enabling 328
parallel processing, implementing 329
RESVSOC
impact on MAXSOC 306
security
asynchronous hold queue 183
connections between IMS Connect instances 181
error response to sample exit RACROUTE calls 185
IMS Universal drivers 179
OTMA Resume TPIPE Security user exit (OTMARTUX) 183
overview 156
PassTicket, RACF 193
RACF PassTicket 194
Resume tpipe 183
specifying OTMA ACEE aging value 199
trusted users 198
user message exit routines 161
security exit routine 183
security for 183
security, default RACF user ID 178
send-only protocol
synchronous callout, coding for 207
Send-only protocol
with acknowledgment 297
with error 298
with serial delivery 298
IMS Connect *(continued)*
send-only protocol and callout requests 299
send-then-commit timeout value 283
shared queues and ALTPCB output 174
SOA composite business application support 285
SOAP Gateway 855
socket connections
non-persistent 301
persistent 300
transaction 301
sockets
maximum number 304
processing for transactions 303
reserving 306
reset threshold 307
setting percentages for warnings 307
UNIX System Services maximum number 305
warning threshold 307
super member
IMS-to-IMS TCP/IP communications 153
Sysplex Distributor 155
TCP/IP
failures 309
KeepAlive intervals 309
TCP/IP communications 139
TCP/IP settings 361
timeout intervals, setting 311
timeout specifications
IMS DB clients 310
IMS TM clients 310
IMS-to-IMS TCP/IP connections 320
timeout value, CM1 ACK/NAK 283
timer
canceling 319
transaction expiration
setting time in IRM 322
transaction protocols 283
transactions
restrictions 283
two-phase commit
cross-LPAR support for IMS TM transactions 355
IMS TM Resource Adapter support 352
IMS Universal drivers support 346
Unicode 359
UNIX System Services
socket limits 305
user ID, default for RACF 178
user message exit routines
overview 160
security 161
user message exits
output message structure 229
user-written client application
message structures 233
XIB (exit interface block)
format of entry 164
format, IMS DB connections 168
XIB1 format 168
XIB1
format 168
XIBDS

IMS Connect *(continued)*
XIBDS *(continued)*
format of entry 165
XIBOD
format of entry 169
XML
COBOL conversion example 212
converting to COBOL 209
message structures 210
XML adapter 154
XML conversion support, overview 154
XML converter 154
XML converters 210
z/OS Sysplex Distributor 155
IMS message switches, response requirements 947
IMS Monitor
MSC considerations 710
Report Print Program 711
IMS services available to ESAP 119
IMS TM
IMS Connect
timeout specifications 310
IMS TM Resource Adapter
calling out to external services from IMS 853
IMS Connect
duplicate client IDs 146
generated client IDs 146
support overview 146
two-phase commit support 352
IMS Connect callout support 855
IMS Connect message structures 233
one-phase commit 356
two-phase commit
IMS Connect support 352
IMS TM resources
resource type consistency
disabling enforcement 380
IMS Universal Database resource adapter
one-phase commit 349
type-2
installing in WebSphere Application Server for z/OS 8
type-4
EAR file, installing on WebSphere Application Server 7
WebSphere Application Server for z/OS
application, installing 11
classpath, setting 9
data source, installing 10
IMS Universal Database resource adapter, installing 5
IMS Universal Database resource adapters
two-phase commit 346
type-2 connectivity
WebSphere Application Server for z/OS configuration overview 7
IMS Universal drivers
CICS
installing on CICS 20
configuring connections 3
IMS Connect
timeout specifications 310
two-phase commit support 346
IMS Connect security 179
IMS Universal drivers *(continued)*
IMS Connect support
connection routing 146
one-phase commit 349
security
IMS Connect 179
two-phase commit
IMS Connect support 346
WebSphere Application Server
configuring 4
IMS-CICS communication
alternate facility 625
application coding for 622
asynchronous processing flow 609
ATTACH parameters 629
CICS transactions, definition 620
coding
function management headers 629
system definition options 613
facility
alternate 625
IMS commands 626
IMS-CICS ISC 620
initiating sessions 620
integrity of session 637
LU 6.1 links
compatible nodes 614
description 613
Macro-Level Resource Definition 613
multiple links 618
Resource Definition Online 613
MFS support 634
preparing CICS tables 613
principal facility 625
processing flows
RECEIVE 608
RETRIEVE 612
SEND INVITE 605
SEND LAST 607
SEND/RECEIVE 605
START/RETRIEVE 610
recovery and restart 635
SCHEDULER parameters 632
session
binding 637
initiation 620
processing outstanding traffic 638
reestablishing 637
resynchronizing 637
sync points 627
termination 621
sync points 627
terminating a session 621
transactions
attributes supported 589, 590, 625
types supported 589, 590, 625
IMS-to-IMS communication, LU 6.1 protocols 478, 480
IMS-to-IMS sessions, ISC protocol restrictions 480
IMS-to-IMS TCP/IP communications
IMS Connect
timeout specifications 320
OTMA
format of output messages 780

IMS-to-IMS TCP/IP communications *(continued)*
OTMA *(continued)*
IMS-to-IMS TCP/IP communications 780
messages, format on output 780
overview of OTMA super member support 779
overview of OTMA support 778
socket connections
cleanup 302
persistence 302
termination scenarios 302
IMS-to-IMS TCP/IP
connections
RESVSOC parameter
usage 306
socket, reserving 306
IMS.FORMAT, output from MFS 426
IMS.PROCLIB 408
IMSIDs
when an IMSplex and MSC network coexist 719
IMSplex
affinity
in an MSC-IMSplex configuration 715
APPC and OTMA messages
processing MSC remote transactions 719
definition 373
environment requirements 173
IBM Management Console for IMS and Db2 for z/OS 173
IMS Connect configuration file 174
IMS Connect support 174
IMSIDs
when an IMSplex and MSC network coexist 719
installation 174
link definitions in an IMSplex
deleting 719
message routing
in an MSC-IMSplex configuration 713
MSC (Multiple Systems Coupling)
coexistence 713
migration 715
sharing MSNAME definitions and SYSIDs 716
MSC MSNAME definitions
duplication 718
sharing MSNAME definitions and SYSIDs 716
MSNAME definitions
deleting 718
OM access 173
pseudoabend U0830
avoiding 725
SCI (Structured Call Interface) 173
SYSIDs
cloning MSC SYSIDs in an IMSplex 719
managing MSC SYSIDs in an IMSplex 719
sharing MSNAME definitions and SYSIDs 716
terminal management 378
IMSplex affinity routing 732
IMSplex support 173
IMSplexes
MSC
APPC and OTMA remote transactions 719
IMSplexes *(continued)*
MSC *(continued)*
back-end processing of remote APPC or OTMA transactions 720
IMSRSC repository
usage with MSC 733, 734, 737, 738
Information Management System (IMS) 368
initiation
session 420
INOP state 396
input bracketing
replies 936
input component
identification for header type X'42' 935
relationship to output component 462
versus output component 907
input editing option
ISC 459
SLU P 916
input errors, IMS detection of ISC 535
input message
backing out ISC 507
bracketing 933, 936
chained messages 524, 936
Fast Path 919
Finance Communication System chaining indicator 933
header
DPM 934
Finance format 933
MID name location 933, 934
optional MFS (Message Format Service) fields 934
SCS 934
SLU P format 934
use 933
ISC bracketing 515
message descriptor byte
header type X'42' 935
MID name indicator bit 934, 935
MFS (Message Format Service) input formatting, activating 936
MFS input formatting, activating 547
multiple transmissions for one message 936
origin of for MFS (Message Format Service) purposes 936
requirements 933
response requirements 944
SLU P
chaining indicator 933
restriction 936
sync point
availability 507
ISC 507
input message structure
from clients 231
input message, response requirements 510
input system (MSC) 676
Input/Output Edit exit routine (DFSYIOE0) 781
inquiry flag processing 118
integrity
message
IMS-CICS session 637
in ISC 449
NOCHECK PROTECT 610
interchange unit code, ATTACH FM header 559

intermediate IMS
MSC
security checking 705
intermediate system (MSC) 676
internal execution mode, ISC 462
Intersystem Communication (ISC)
application program
accessing 468
example 657
not using MFS 469
routing example 471, 473–477
ATTACH FM header
bit contents 557
format 557
ATTDBA 560
ATTDSP 560
binding an ISC session 490
CICS
abends, transaction 639
asynchronous transactions, defining 620
ATTACC 631
ATTDBA 631
ATTDP 630
ATTDPN 629
ATTDQN 630
ATTDSP 631
ATTIU 631
ATTPRN 630
ATTRDPN 630
ATTRPRN 630
data stream profile (ATTDSP) 631
deblocking algorithm (ATTDBA) 631
in-doubt processing 620
interchange unit code (ATTIU) 631
network name 614
number of sessions 615
outbound destination process name field (ATTDPN) 629
outbound destination process name field (SCDDPN) 632
primary resource name field (ATTPRN) 630
primary resource name field (SCDPRN) 632
restarting TCP/IP sessions 604
return destination process name (ATTRDPN) 630
return destination process name (SCDRDPN) 633
return primary resource name (ATTRPRN) 630
return primary resource name (SCDRPRN) 633
sample program DFSISC00 658
sample program, ACB generation 664
sample program, installing 657
sample program, JCL to compile and bind 660
sample program, MFS formats 662
sample program, PSB generation 663
sample program, system definition statements 661
SCDDP 633
SCDDPN 632
SCDDQN 633
SCDPRN 632
SCDRDPN 633
SCDRPRN 633
session initiation 621
session names 615
session parameters 615
session restart, TCP/IP 604
Intersystem Communication (ISC) *(continued)*
CICS *(continued)*
session termination, abnormal, TCP/IP 604
session termination, orderly, TCP/IP 603
session termination, TCP/IP 603
session termination, TCP/IP unconditional 604
TCP/IP session restart 604
TCP/IP session termination 603
TCP/IP session termination, abnormal 604
TCP/IP session termination, orderly 603
TCP/IP session termination, unconditional 604
terminal device-dependent data 453
terminating TCP/IP sessions 603
terminating TCP/IP sessions unconditionally 604
terminating TCP/IP sessions, orderly 603
transaction abends 639
CICS-IMS
application-related concepts 622
control block storage, parallel sessions 461
conversation mode termination extension 506
data descriptor FM header
use on input 551
data flow control
example 643, 644, 648, 649, 651
DFC protocols 515
DFC sync point
exception 507
irrecoverable messages 510
MFS messages 508
on output 508
recoverable messages 508
editing facilities
invoking FM headers 542
overview 459
editing options 435
error handling
MTO notified 532
sender ERP 536
error recovery procedure
examples 653
examples, receiver-detected errors 654
examples, sender-detected errors 653
error recovery, purge after exception response 525
ETO 61
examples of ATTACH and SCHEDULER parameters with MFS 581
examples of ATTACH parameters with SYSMSG 579
execution mode
external specification 461
facilities 459
FM headers
ATTDBA 560
ATTDSP 560
CICS session 629
format reference 557
introduction 541
MFS 547
QGET 567
QGETN 568
QPURGE 569
QSTATUS 570
QXFR 571
RAP 573
routing messages 469

Intersystem Communication (ISC) *(continued)*
FM headers *(continued)*
SCHEDULER 573
supported by IMS 541
SYSERROR 575
SYSSTAT 575
half session, primary versus secondary 490
IMS commands, issuing 460
IMS support, CICS 587
IMS-to-IMS session
design considerations 478
MSC and ISC coexistence 478
interchange unit code, ATTACH FM header 559
introduction 445
ISC edit ATTACH parameter examples 577
ISC node
defining 613
ISC protocol
communications design 467
ISC sessions
unsupported commands 612
message integrity, CICS 636
message resynchronization
sessions 491
message routing
FM headers 469
message switch
ATTACH FM header 557
MSC coexistence 478
multichain input message switches restriction 559
network operation 487
network, bringing up 487
network, starting 487
node definition
macros used 480
output editing option 459
output protocols 467
parallel sessions issuing IMS commands 460
protocols
defined 487
restrictions on IMS-to-IMS sessions 480
QGET 567
QGETN 568
QPURGE 569
QSTATUS 570
QXFR 571
RAP 573
resynchronization procedure 492
routing examples 471
routing parameters 470
SC (session control) protocols 488
SC protocols
session initiation 488
SCHEDULER 573
selective receiver ERP
X'0864xxxx': function abort 533
X'0865xxxx': function abort 533
X'0866xxxx': function abort 534
sense codes
received 538
statically defining 480
sync-point indicators 510
SYSERROR 575
Intersystem Communication (ISC) *(continued)*
SYSSTAT 575
system definition
macro parameters 482
TCP/IP
ETO 61
IMS Connect, overview of support 148
restarting sessions 604
session restart 604
session termination 603
session termination, abnormal 604
session termination, orderly 603
session termination, unconditional 604
terminating sessions 603
terminating sessions unconditionally 604
terminating sessions, orderly 603
TCP/IP support
CICS front-end transaction types 604
defining the link to CICS 599
defining the link to IMS Connect 598
dynamic terminal definition 596
falling back to VTAM 598
overview 593
requirements 594
restrictions 595
security 595
sessions, starting 601
sessions, starting from CICS 602
static terminal definition 596, 597
switching from TCP/IP to VTAM 598
terminal definition 596
test mode 460
transaction types supported, CICS session 589, 590, 625
versus CICS 625
VTAM facilities 455
XRF
USERVAR, session initiation 489
XRF complex 489
invalid destinations in MSC (Multiple Systems Coupling) 684
IOPCB 758
IRM
IMS Request Message) 227
IRM extensions 227
IRM (IMS request message)
fixed portion format 214
IRM (IMS Request Message) 214, 218, 359
IRM_TIMER
usage 311
values 311
irrecoverable messages, ISC sync points 510
irrecoverable output 825
irrecoverable transactions 479
irrecoverable-inquiry transactions
LU 6.2 application program 421
response requirements 421, 946
ISC
CICS
asynchronous processing, overview 587
functions supported 587
synchronous processing, overview 587
functions supported
CICS sessions 587
TCP/IP

ISC *(continued)*
TCP/IP *(continued)*
asynchronous processing, overview 587
functions supported 587
synchronous processing, overview 587
ISC (Intersystem Communication)
application program
accessing 468
example 657
not using MFS 469
routing example 471, 473–477
using MFS (Message Format Service) 468
ATTACH FM header
bit contents 557
format 557
ATTDBA 560
ATTDSP 560
basic functions 445
binding an ISC session 490
CICS
abends, transaction 639
asynchronous transactions, defining 620
ATTACC 631
ATTDBA 631
ATTDP 630
ATTDPN 629
ATTDQN 630
ATTDSP 631
ATTIU 631
ATTPRN 630
ATTRDPN 630
ATTRPRN 630
data stream profile (ATTDSP) 631
deblocking algorithm (ATTDBA) 631
in-doubt processing 620
interchange unit code (ATTIU) 631
network name 614
number of sessions 615
outbound destination process name field (ATTDPN) 629
outbound destination process name field (SCDDPN) 632
primary resource name field (ATTPRN) 630
primary resource name field (SCDPRN) 632
restarting TCP/IP sessions 604
return destination process name (ATTRDPN) 630
return destination process name (SCDRDPN) 633
return primary resource name (ATTRPRN) 630
return primary resource name (SCDRPRN) 633
sample program DFSISC00 658
sample program, ACB generation 664
sample program, installing 657
sample program, JCL to compile and bind 660
sample program, MFS formats 662
sample program, PSB generation 663
sample program, system definition statements 661
SCDDP 633
SCDDPN 632
SCDDQN 633
SCDPRN 632
SCDRDPN 633
SCDRPRN 633
session initiation 621
session names 615
ISC (Intersystem Communication) *(continued)*
CICS *(continued)*
session parameters 615
session restart, TCP/IP 604
session termination, abnormal, TCP/IP 604
session termination, orderly, TCP/IP 603
session termination, TCP/IP 603
session termination, TCP/IP unconditional 604
TCP/IP session restart 604
TCP/IP session termination 603
TCP/IP session termination, abnormal 604
TCP/IP session termination, orderly 603
TCP/IP session termination, unconditional 604
terminal device-dependent data 453
terminating TCP/IP sessions 603
terminating TCP/IP sessions unconditionally 604
terminating TCP/IP sessions, orderly 603
transaction abends 639
CICS and IMS functions 452
CICS-IMS
application-related concepts 622
Class 2 terminal 449
configurations 450
control block storage, parallel sessions 461
conversation mode termination extension 506
data descriptor FM header
use on input 551
data flow control
example 643, 644, 648, 649, 651
DFC protocols
availability 507
backing out input messages 507
BID command 515
bracket and half-duplex 515
bracketing input messages 515
bracketing output messages 518
CANCEL command 523
chaining 524
CHASE command 524
conversation mode errors 506
exception response 505
half-session synchronization 506
LUSTATUS 528
paged message errors 531
recoverability supplemented by LWA 506
response mode errors 505
response requirements 506
RSHUT command 531
RTR command 531
selective receiver ERP 531
sync point, input messages 507
DFC sync point
exception 507
irrecoverable messages 510
MFS messages 508
on output 508
recoverable messages 508
distributed processing 447
distributed services, support for 448
editing facilities
invoking FM headers 542
overview 459
editing options 435
error handling

ISC (Intersystem Communication) *(continued)*
error handling *(continued)*
MTO notified 532
sender ERP 536
error recovery procedure
examples 653
examples, receiver-detected errors 654
examples, sender-detected errors 653
error recovery, purge after exception response 525
ETO 61
examples of ATTACH and SCHEDULER parameters with MFS 581
examples of ATTACH parameters with SYSMSG 579
execution mode
external specification 461
internal definition 462
internal, with CICS 589, 590
processing mode 462
synchronous versus asynchronous 462
facilities 459
FM headers
ATTACH 543
ATTACH, MFS 550
ATTDBA 560
ATTDSP 560
CICS session 629
data descriptor 551
DPM messages, MFS 548
error recovery procedure 543
format reference 557
initiating a process 543
introduction 541
invoking ISC edit 542
MFS 547
QGET 567
QGETN 568
QMODEL 552
QPURGE 569
QSTATUS 570
QXFR 571
RAP 573
RAP (reset attached process) 544
RAP (reset attached process), MFS 555
reply QMODEL 554
routing messages 469
SCHEDULER 544, 573
SCHEDULER, MFS 550
supported by IMS 541
synchronous processing 543
SYSERROR 575
SYSMSG 545
SYSSTAT 575
half session, primary versus secondary 490
IMS commands, issuing 460
IMS facilities 447
IMS support for 447
IMS support, CICS 587
IMS-to-CICS configuration 450
IMS-to-IMS configuration 451
IMS-to-IMS session
buffer sizes 479
defining parallel session 478
defining single session 478
dequeuing messages 494
ISC (Intersystem Communication) *(continued)*
IMS-to-IMS session *(continued)*
design considerations 478, 480
MSC and ISC coexistence 478
protocol restrictions 480
remote control 479
restriction on conversation mode 479
routing to back-end IMS 479
interchange unit code, ATTACH FM header 559
introduction 445
ISC edit ATTACH parameter examples 577
ISC node
defining 613
ISC protocol
communications design 467
ISC sessions
unsupported commands 612
message integrity 449
message integrity, CICS 636
message resynchronization
sessions 491
summary 512
message routing
FM headers 469
parameters 469, 470
through MSC links 478
message switch
ATTACH FM header 557
MSC coexistence 478
multichain input message switches restriction 559
network operation 487
network, bringing up 487
network, starting 487
node definition
COMM macro 482
example 480
macros used 480
NAME macro 483
SUBPOOL macro 483
summary 485
TERMINAL macro 483
output editing option 459
output protocols 467
parallel sessions issuing IMS commands 460
passing CICS data to IMS 453
protocols
defined 487
restrictions on IMS-to-IMS sessions 480
summary 445
QGET 567
QGETN 568
QPURGE 569
QSTATUS 570
QXFR 571
RAP 573
resynchronization procedure
cold start, to recover sessions 493
commands, recovery at failure 495
controlling backout of work unit 493
execution mode, recovery at failure 495
maintaining sequence numbers 493
output available at restart 494
recovering from in-bracket failure 494

ISC (Intersystem Communication) *(continued)*
resynchronization procedure *(continued)*
restart process 495
session failure, IMS failure 495
session failure, IMS still running 495
routing 445
routing examples 471
routing parameters 470
SC (session control) protocols 488
SC protocols
binding sessions 490
cold start recovery 493
commands, recovery at failure 495
controlling pending work units 493
designing restart procedures 492
execution mode, recovery at failure 495
maintaining sequence numbers 493
output available at restart 494
recovering from in-bracket failure 494
resolving bind race 491
restart process 495
running the session 499
session failure, IMS failure 495
session failure, IMS still running 495
session initiation 488
session states 498
session termination 499, 536
XRF, session initiation 489
SCHEDULER 573
security 449
selective receiver ERP
X'0864xxxx': function abort 533
X'0865xxxx': function abort 533
X'0866xxxx': function abort 534
sense codes
received 538
sent 539
session control 445
statically defining 480
sync-point indicators 510
SYSERROR 575
SYSSTAT 575
system definition
macro parameters 482
TCP/IP
ETO 61
IMS Connect, overview of support 148
restarting sessions 604
session restart 604
session termination 603
session termination, abnormal 604
session termination, orderly 603
session termination, unconditional 604
terminating sessions 603
terminating sessions orderly 603
terminating sessions unconditionally 604
TCP/IP support
CICS front-end transaction types 604
defining the link to CICS 599
defining the link to IMS Connect 598
dynamic terminal definition 596
falling back 598
overview 593
requirements 594
ISC (Intersystem Communication) *(continued)*
TCP/IP support *(continued)*
restrictions 595
security 595
sessions, starting 601
sessions, starting from CICS 602
static terminal definition 596, 597
switching from TCP/IP to VTAM 598
terminal definition 596
test mode 460
transaction types supported, CICS session 589, 590, 625
versus CICS 625
VTAM API
CICS 589, 590
IMS 457
VTAM facilities 455
XRF
MNPS, session initiation 489
USERVAR, session initiation 489
XRF complex 449, 489
ISC data flow control
examples 643
ISC edit
default editor 459
non-MFS programs 469
ISC-CICS installation options 613

J

Java
accessing Db2 for z/OS data from IMS 111
connecting to IMS from external Java environments 1
Db2 for z/OS
accessing from Java dependent regions 111
dependent regions
accessing Db2 for z/OS
111
Java application programs
calling out to external services from IMS 853
callout support overview 855
JBPs
accessing Db2 for z/OS data from IMS
111
JCL
to protect transactions 836
JDBC
type-2
IMS Universal drivers and CICS 19
IMS Universal JDBC driver and CICS 19
JMPs
accessing Db2 for z/OS data from IMS
111
JOBLIB
TM and MSC Message Routing and Control user exit routine 730

K

keyboard shortcuts xxv

L

language interface 134
language interface entry points 134
language interface module 134
legal notices
  notices 953
  trademarks 953, 954
libraries, online change 432
link
  MSC (Multiple Systems Coupling) 667
  priorities, setting 699
link paths
  MSC (Multiple Systems Coupling) 670
Link-Receive Routing exit routine in MSC 732
LINKLIST
  TM and MSC Message Routing and Control user exit routine 730
links
  bandwidth of MSC links 686
  deleting MSC link definitions in an IMSplex 719
  MSC
    optimum link types 706
  MSC (Multiple Systems Coupling)
    definitions in an IMSplex 715
    logical 669
    physical 667
  MSC bandwidth mode, capacity 707
  MSC logical link benchmarks 706
  MSC logical link capacity 707
  MSC logical link high value statistics 710
  MSC logical link statistics 705
  MSC logical link statistics and buffer sizes 707
  MSC logical link statistics, resetting 706
  MSC, controlling bandwidth 686
list structure
  definition 374
  overflow, definition 375
  primary, definition 375
loading ESAP 129
loading external subsystem modules 130
local option client communication 142
local system
  MSC (Multiple Systems Coupling)
    defined 671
lock mode 395
LOCK state 422
Log Analysis report
  ID column for MSC entries 712
  MSC transactions 712
Log Merge utility
  log output, control of 712
  MSC logs 712
Log Transaction Analysis utility, MSC statistics 712
log write-ahead (LWA) 449
logging
  MSC 711
logging off 372
logging off, definition 372
logging on, definition 372
logical destinations
  MSC (Multiple Systems Coupling) 674
logical link paths
  MSC (Multiple Systems Coupling) 670
logical links
  defining 698
  MSC (Multiple Systems Coupling) 669
logical network design 390
logical page advance function (NEXTLP), MFS 913
logical page requests function (PAGE=nn), MFS 913
logical terminal (LTERM)
  chains 391
  component definition 560
  convention for naming
    eliminating the need for 908
  definition 385
  ETO 369
  master terminal 393
  method for naming 907
  MSC and 732
  network design 390
  queues 392
  relationship to physical terminals 390
  remote 681
  separating input and output devices 392
  subpools, users and components 463
logical unit
  definition 366
  programmable, definition 367
logical unit (LU)
  multiple, managing 53
  qualifying LU names 52
  reassigning 53
logical unit of work 636
logical unit status command 528
logon descriptor
  ETO
    definition 63
logon mode
  default logon mode table 457
  session initiation 904
LOGON MODE table 418
LTERM (logical terminal)
  chains 391
  component definition 560
  convention for naming
    eliminating the need for 908
  definition 385
  ETO 369
  master terminal 393
  method for naming 907
  network design 390
  queues 392
  relationship to physical terminals 390
  remote 681, 732
  separating input and output devices 392
  subpools, users and components 463
LU
  APPC outbound LU specification 42
  outbound
    specifying 42
LU (logical unit)
  multiple, managing 53
  qualifying LU names 52
  reassigning 53
LU 6.2
  descriptors 385
LUSTATUS command

LUSTATUS command *(continued)*
CICS 592, 625
commit 530
conversation mode
requesting normal termination 506
sending errors 506
function abort 529
NO-OP 530
paging errors 531
protocol 528
queue empty 529
response mode errors 505
SLU P system 950
terminating test mode 460
LWA (log write-ahead) 506

M

M descriptors, OTMA
limit 778
maximum 778
macro keywords
APPLID on COMM macro 416
COMM on BUFPOOLS macro 417
MODETBL on ETO logon descriptor 418
MODETBL on TERMINAL macro 418
PASSWD on COMM macro 416
RECANY on COMM macro 417
macros
IMS Connect 162
XCF, and OTMA 791
maintenance 431
making CICS ready 620
making IMS ready 487, 923
master terminal
device choice 393
logon requirements 418
MSC routing 732
reserving an LTERM 385
XRF complex 393
master terminal operator (MTO) 418
MAXFILEPROC parameter, UNIX System Services 305
memory-to-memory (MTM)
MSC (Multiple Systems Coupling) physical link type 667
MSC physical link
defining 696
message
architected form 824
bypassing MFS editing 436
control characters 438
destination determination 782
editing of output segments 437
editing performed by IMS 434
examples of how to select 785
extending 756
flow
commit-then-send 826, 834
deallocate 824
definition 757
in full-duplex environment 764
resynchronization 842
send-then-commit 829
send-then-commit with Confirm 830
use of queues in tpipe 763
message *(continued)*
flow in an OTMA environment 758
formatting and editing 432
full-duplex flow 759
half-duplex flow 759
IMS handling 373
in-flight
device LUs 419
program LUs 419
output segment editing 437
prefix
contents 834
OTMA application-data section 886
OTMA security data 883
OTMA security-data section 883
OTMA user data section 886
state-data section 868
syntax 859
requeuer 785
resynchronization
sample 844
sample
acknowledge receipt of CBresynch 844
acknowledge receipt of SRVresynch 844
client-bid request with resynchronization 844
REPresynch command 845
REQresynch command 845
SRVresynch command 844
successful resynchronization 845
sample OTMA message 887
scheduling
definition 387
Fast Path 380
selective recovery 785
sensitivity to nongraphic message data 437
sequence numbers 823
sequential order 830
switch
in shared queues environment 820
translation 359
Z2 field 438
message advance function (NEXTMSG) 913
message advance protect function (NEXTMSGP) 913
message buffering with a Fast Path-capable system 381
message contention 491, 917
Message Control/Error exit routine (DFSCMUX0) 702
message descriptor byte
input FM header 933, 935
output message 939, 941
message format
OTMA
IMS-to-IMS TCP/IP communications 780
Message Format Service
OTMA, unsupported 791
Message Format Service (MFS)
administration 430
advantages 431
application programs, accessing with ISC 468
Bid options 914
bypassing MFS 436
bypassing with ISC application programs 468
components 426
components, overview 432

Message Format Service (MFS) *(continued)*
control functions
Finance Communication System 913
NEXTLP 913
NEXTPP 913
PAGE=nn 913
control requests, response requirements 947
data bytes output message 942
default format 575
defining 911
delete characters 437
description 370
DPM 939
DPM option 437
edit of ISC messages 459
editing 436
escape characters, not supported in ISC 547
facilities available 911
FM header
activating output formatting 547
FM headers
activating input formatting 547
editing 547
facilities available 547
types 547
formatting
activating input 924
input formatting 547
input messages 425
input segments 437
introduction 425
invalid paging request 915, 917
Language utility 426, 432
libraries, online change 432
message editor 433
message recovery 913
MID/MOD chaining 912
MSC (Multiple Systems Coupling) 689
online error detection 548
online performance 431
output formatting
MOD name 943
typical application program procedure 943
output formatting, ISC 547
output messages
editing segments 437
how MFS defines 425
overview 425
page delete function, not supported in ISC 548
paging, CICS 590, 624
pool manager 433
SDF II 433
Service utility 432
SLU P
benefits 912
station-by-station, availability 912
sync point, ISC messages 508
terminal keyboard lock and unlock 436
message handling 373
message headers 437
message input descriptor (MID) 432
message integrity 449
message output descriptor (MOD) 432, 943
message queue, definition 380
message recovery
Finance Communication System 918
message resynchronization 512, 918
MFS 913
restriction 919
SLU P system 918
message resynchronization
associated system definition options 492
CICS-IMS session 637
description 492, 637
design considerations 492, 925
direction and bracket indicators 932
effects on normal data transmission 918, 925
Fast Path 921
how and when initiated 491, 918
last inbound/outbound 918
options for message sequence numbers 925
performing 637
polarity of half-session pairs 492
purpose 491
requirements 491, 918
send/receive and bracket protocol 932
sequence numbers 493, 925
STSN
flow, primary-to-secondary 500
flow, secondary-to-primary 502
format 503, 926
use of 496
message routing
examples 471
in an MSC (Multiple Systems Coupling) network 673
in ISC
across an MSC link 478
use of routing parameters 470
message structures
for IMS TM Resource Adapter 233
for user-written IMS Connect client application programs 233
message switch
ATTACH FM header 557
examples 471
ISC 469, 470
message switch, ISC 459
message switches, response requirements 510
message switching, DFSAPPC 54
messages
affinity routing 732
flow
using transaction pipes 763
using XCF 763
IMSplex 730
IMSplex affinity routing 732
MSC 730
OTMA
sending 822
protected conversation, OTMA restrictions 791
routing
IMS-to-IMS TCP/IP communications flow 759
in an MSC-IMSplex configuration 713
OTMA destination descriptor 774
OTMA destination descriptor, defining 776
OTMA destination descriptor, masking destination names 777

messages *(continued)*
routing *(continued)*
TM and MSC Message Routing and Control user exit routine 730
terminal/input routing 730
TM and MSC Message Routing and Control user exit routine 730
MFS (Message Format Service)
administration 430
advantages 431
application programs, accessing with ISC 468
Bid options 914
bypassing MFS 436
bypassing with ISC application programs 468
components 426
components, overview 432
control functions
Finance Communication System 913
NEXTLP 913
NEXTMSG 913
NEXTMSGP 913
NEXTPP 913
PAGE=nn 913
SLU P 913
control requests, response requirements 947
data bytes output message 942
default format 575
defining 911
delete characters 437
description 370
DPM 939
DPM option 437
edit of ISC messages 459
editing 436
escape characters, not supported in ISC 547
facilities available 911
FM headers
activating input formatting 547
activating output formatting 547
ATTACH 550
data descriptor 551
DPM messages 548
editing 547
facilities available 547
MOD name 547
QGET 553
QGETN 553
QMODEL 552, 634
QPURGE 554
QSTATUS 554
QXFR 554
RAP (reset attached process), MFS 555
reply QMODEL 554
SCHEDULER 550
specifying MID name 547
types 547
formatting
activating input 924
activating output 936
input formatting 547
input messages 425
input segments 437
introduction 425
invalid paging request 915, 917
MFS (Message Format Service) *(continued)*
Language utility 426, 432
libraries, online change 432
message editor 433
message recovery 913
MID/MOD chaining 912
MSC (Multiple Systems Coupling) 689
online error detection 548
online performance 431
output formatting
MOD name 943
typical application program procedure 943
output formatting, ISC 547
output messages
editing segments 437
how MFS defines 425
overview 425
page delete function, not supported in ISC 548
paging, CICS 590, 624
pool manager 433
SDF II 433
Service utility 432
SLU P
benefits 912
station-by-station, availability 912
sync point, ISC messages 508
terminal keyboard lock and unlock 436
MFS bypass option effect 426
MFS device characteristics table
entries 383
how used 75
use with non-SNA 3270 devices 76
MFS Distributed Presentation Management (DPM) 459
MFS Service utility 429
MFSTEST procedure 433
MID (message input descriptor) 432
MID/MOD chaining 912
migration
IMSplex from MSC 715
MSC to IMSplex 715
mirror transaction, CICS 609
mixed-case passwords
IMS Connect 192
MNPS (multinode persistent sessions)
XRF 406
MOD (message output descriptor)
name specification 547, 943
purpose 432
modes, terminal
conversation 394
ETO and exclusive mode 395
exclusive 395
lock 395
LU 6.2 394
response 394
SNA QUIESCE 396
test mode 395
MODETBL
specifying a default LOGON MODE identifier 418
MODETBL= keyword
overriding the defaults 457
use during ISC logon 457
modified application program

modified application program *(continued)*
LU 6.2 descriptor 45
remote execution, MSC 38
modules
DFSAERA0 748
DFSAERG0 748
DFSAERM0 748
DFSCDLI0 748
monitoring and tuning, MSC 710
monitoring performance 792
MSASSIGN (/MSASSIGN) command
assigning physical link to logical link 669
MSC
message routing 713
MSC (Multiple Systems Coupling)
/MSVERIFY command 704
administration
APPC 685
affinity
in an MSC-IMSplex configuration 715
affinity management
TCP/IP generic resources 726
affinity persistence
TCP/IP generic resources 728
affinity, clearing
IMS Connect 729
TCP/IP generic resources 728
APPC and OTMA messages
remote processing 719
APPC/IMS
application program failure and transaction recovery 50
intermediate IMS failure and transaction recovery 50
local IMS failure and transaction recovery 49
local transaction discardability 47
LU 6.2 recoverability flows 50
LU 6.2 sessions and transaction recovery 48
MSC links and transaction recovery 49
remote IMS failure and transaction recovery 49
transaction point of failure 48
bandwidth mode, capacity 707
bandwidth, link 686
benchmarking logical link performance 706
buffer sizes 695
buffer sizes, links 686
buffers
format in bandwidth mode 708
format in non-bandwidth mode 709
size in bandwidth mode 708
size in non-bandwidth mode 709
size, optimizing with link statistics 707
coexistence with ISC 478
comparison to ISC 446
concepts 667
data flow 672
data partitioning 667
defining priorities 699
defining SYSIDs 693
definition 667
definition of transaction codes 689
design considerations 685
destination system 676
MSC (Multiple Systems Coupling) *(continued)*
directed routing, program-to-program switch 684
disabling 693
enabling
system definition, during 692
enabling for DRD 690
enabling for IMSRSC repository 691
functions compared to ISC 446
generic resources
affinity management, TCP/IP links 726
affinity persistence, TCP/IP links 728
affinity, clearing for TCP/IP links 728
affinity, clearing in IMS Connect 729
Generic Resources
TCP/IP 726
Generic Resources, VTAM 730
IMS Connect
generic resources 152
IMS-to-IMS TCP/IP communications
IMS Connect support for MSC 151
IMSIDs
in an IMSplex 719
IMSplex
sharing MSNAME definitions and SYSIDs 716
IMSplex with shared queues
coexistence 713
IMSplexes
APPC and OTMA remote transactions 719
APPC transactions, sending to an MSC system outside of an IMSplex 725
back-end processing of remote APPC or OTMA transactions 720
enabling back-end processing of remote APPC and OTMA transactions 724
message routing in an IMSplex 714
message routing when an IMSplex and MSC coexist 714
OTMA transactions, sending to an MSC system outside of an IMSplex 725
input system 676
intermediate system 676
introduction 667
ISC facility 446
link bandwidth 686
link buffer sizes 686
link definitions
in an IMSplex 715
link definitions in an IMSplex
deleting 719
link statistics
assessing optimum link buffer size 707
link type, optimum 706
links
logical 669
physical 667
local system
defined 671
logical destinations 674
logical link capacity 707
logical link paths 670
logical link statistics 705
logical link statistics and buffer sizes 707
logical link statistics, high values 710
logical links

MSC (Multiple Systems Coupling) *(continued)*
logical links *(continued)*
benchmarking 706
buffer sizes and statistics 707
capacity and statistics 707
deleting from an IMSplex 719
high value statistics 710
resetting statistics 706
statistics and buffer sizes 707
statistics and link capacity 707
statistics for benchmarking 706
statistics, high values 710
statistics, resetting 706
logical links, SDLC link 669
LU 6.2 application transactions buffer size 695
message routing
in an MSC-IMSplex configuration 713
migration to IMSplex 715
minimizing resource consumption 685
monitoring and tuning 710
MSC conversation failure 689
MSNAME definitions
sharing in an IMSplex 716
MSNAME definitions in an IMSplex
deleting 718
MSNAME duplication in an IMSplex with shared queues 718
operating procedures 705
optimum link type 706
overview 370
overview of network and routing 671
performance 705, 711
physical links
CTC, defining 696
defining 694, 695
deleting from an IMSplex 719
MTM, defining 696
TCP/IP, defining 697
types 694
VTAM, defining 697
pseudoabend U0830
avoiding 725
Queuing Summary Report 711
recoverable versus irrecoverable transactions 47
recovering transactions in APPC 47
remote system
defined 671
resetting logical link statistics 706
routing path 673
security checking
intermediate IMS 705
security considerations 705
serial transaction processing 701
shared queues
MSNAME duplication 718
sharing MSNAME definitions and SYSIDs 716
SPA 688
standard application programs 36
statistics
assessing optimum link buffer size 707
statistics, for logical links 705
support for APPC/IMS 46
SYSID (system identifier) 677
MSC (Multiple Systems Coupling) *(continued)*
SYSID tables
deleting MSNAME definitions in an IMSplex 718
SYSIDs
cloning in an IMSplex 719
managing in an IMSplex 719
sharing in an IMSplex 716
system definition 689
system definition verification 704
system identifier (SYSID) 677
TCP/IP
generic resources 726
TCP/IP generic resources
affinity management 726
affinity persistence 728
affinity, clearing 728
affinity, clearing in IMS Connect 729
tuning links 705
usage of the IMSRSC repository 733, 734, 737, 738
utility for verifying names 703
VTAM Generic Resources 730
MSC Program Routing exit routine 732
MSC=
disabling MSC 693
enabling MSC 691
MSDB
OTMA, read only access 792
MSLINK definitions
deleting from an IMSplex 719
MSLINK macro
defining 698
MSNAME definitions
deleting from an IMSplex 718
in a shared queues group
duplication 718
sharing among systems in an IMSplex 716
MSNAME macro
logical link paths 670
logical path name 698
MSC (Multiple Systems Coupling) Directed Routing 683
reports
for MSC 711
SYSID keyword 698
MSPLINK definitions
deleting from an IMSplex 719
MSPLINK macro
defining CTC physical links 696
defining MTM physical links 696
defining physical links
buffer sizes 695
defining TCP/IP physical links 697
defining VTAM physical links 697
MTM (memory-to-memory)
MSC (Multiple Systems Coupling) physical link type 667
MSC physical link
defining 696
MTO (master terminal operator)
/OPNDST command 418
assignment of logical link 667
ISC errors 532
notification of session rejection 499
restarting the terminal 436
MULT1 parameter 467, 519
MULT2 parameter 468, 519

multichain input message switches restriction 559
multinode persistent sessions (MNPS)
XRF 406
multiple external subsystems 135
multiple signons
description 91
naming conventions for 91
sysplex environment, in 91
multiple systems 685
Multiple Systems Coupling (MSC)
/MSVERIFY command 704
affinity management
TCP/IP generic resources 726
affinity persistence
TCP/IP generic resources 728
affinity, clearing
IMS Connect 729
TCP/IP generic resources 728
bandwidth mode, capacity 707
bandwidth, link 686
benchmarking logical link performance 706
buffer sizes 695
buffer sizes, links 686
buffers
format in bandwidth mode 708
format in non-bandwidth mode 709
size in bandwidth mode 708
size in non-bandwidth mode 709
size, optimizing with link statistics 707
defining SYSIDs 693
definition of transaction codes 689
disabling 693
enabling
system definition, during 692
enabling for DRD 690
enabling for IMSRSC repository 691
generic resources
affinity management, TCP/IP links 726
affinity persistence, TCP/IP links 728
affinity, clearing for TCP/IP links 728
affinity, clearing in IMS Connect 729
TCP/IP and XRF 729
XRF and TCP/IP 729
Generic Resources
TCP/IP 726
Generic Resources, VTAM 730
IMS Connect
generic resources 152
IMS-to-IMS TCP/IP communications
IMS Connect support for MSC 151
IMSplexes
APPC and OTMA remote transactions 719
APPC transactions, sending to an MSC system outside of an IMSplex 725
back-end processing of remote APPC or OTMA transactions 720
enabling back-end processing of remote APPC and OTMA transactions 724
OTMA transactions, sending to an MSC system outside of an IMSplex 725
link bandwidth 686
link buffer sizes 686
link statistics
assessing optimum link buffer size 707
Multiple Systems Coupling (MSC) *(continued)*
link type, optimum 706
logical link statistics 705
logical link statistics and buffer sizes 707
logical link statistics and link capacity 707
logical link statistics, high values 710
logical links
benchmarking 706
buffer sizes and statistics 707
capacity and statistics 707
high value statistics 710
resetting statistics 706
statistics and buffer sizes 707
statistics and link capacity 707
statistics for benchmarking 706
statistics, high value 710
statistics, high values 710
statistics, resetting 706
optimum link type 706
overview of network and routing 671
performance, logical links 705
physical links
CTC, defining 696
defining 694, 695
MTM, defining 696
TCP/IP, defining 697
types 694
VTAM, defining 697
resetting logical link statistics 706
security considerations 705
SPA 688
statistics
assessing optimum link buffer size 707
statistics, for logical links 705
system definition 689
system definition verification 704
TCP/IP
generic resources 726
TCP/IP generic resources
affinity management 726
affinity persistence 728
affinity, clearing 728
affinity, clearing in IMS Connect 729
XRF 729
tuning links 705
VTAM Generic Resources 730
Multiple Systems Verification utility (DFSUMSV0) 683
Multiple Systems Verification utility, using 703
MULTIRTP
callout messages 858
MULTIRTP parameter
super members 771

N

NAME macro
defining an ISC session 483
for remote logical terminals 670
MSNAME macro 702
used to relate components 464
name uniqueness, resource
disabling enforcement 379
naming conventions 430
NCP (Network Control Program) 368

NCP (Network Control Programs) 367
negotiable bind 490
network
  administration
    activities 383
  APPC/IMS, operating with 372
  communications 365
  defining 415
  definition 415, 416, 702
  design considerations 383
  documenting requirements 383
  logical terminal network design 390
  multiple systems, effect of 702
  nonswitched communications network 393
  operating
    establishing communication 372
    Finance Communication System 899, 923
    IMS-CICS 620, 621
    ISC 487
  operations, preparing for 415
  optional components 365
  planning 383
  shutting down 487, 930
  starting
    Finance Communication System 420, 923
  terminating
    Finance Communication System 929, 931
  VTAM and NCP parameters 415
network architecture models 753
network role 366, 368
network security credential propagation
  IMS Connect 179
network security segments
  format 247
network-qualified LU name 52
network, communications 365
networking 685
NO-OP indicated on LUSTATUS command 530
node definition, ISC 480, 485
node name 384
node user descriptor 85
non-conversational program 820
non-persistent socket connections 301
NONE security level 806
nongraphic message data 437
nonswitched communications network 393
NOSCAN option 939
notify message 116, 133
NRESTART command 440
NTO terminals 394
null output message
  purpose 936
  when not sent 912
  when sent 942

O

OASN (origin application sequence number) 135
ODBA (Open Database Access)
  accessing IMS databases 747
  application programs, binding 748
  APSB security 750
  connections
    configuring 741
ODBA (Open Database Access) *(continued)*
  defining security 748
  interface 747
  modules
    which ones to place in JOBLIB 748
    which ones to place in STEPLIB 748
  overview 747
  RAS security 749
  security
    APSB 750
    RAS 749
  setup 747
one-phase commit
  IMS Connect
    IMS Universal drivers 349
  IMS TM Resource Adapter 356
  IMS Universal Database resource adapter 349
  IMS Universal drivers 349
online change
  DISPLAY MODIFY command 422
  libraries 433
  Online Change utility 432
online performance 431
Open Database
  IMS Connect
    timeout specifications 310
Open Database Access (ODBA)
  accessing IMS databases 747
  application programs, binding 748
  APSB security 750
  connections
    configuring 741
  defining security 748
  interface 747
  modules
    which ones to place in JOBLIB 748
    which ones to place in STEPLIB 748
  overview 747
  RAS security 749
  security
    APSB 750
    RAS 749
  setup 747
open systems interconnection 753
Open Transaction Manager Access
  Callable Interface (C/I)
    introduction to 889
    restrictions 892
    security for 892
  headers 251
  sample messages 887
Open Transaction Manager Access (IMS Open Transaction Manager Access)
  Callable Interface (C/I)
    initializing 891
Open Transaction Manager Access (OTMA)
  ACEE, refreshing 810
  administering 767
  asynchronous hold queues
    security 814
  benefits 755
  callout
    implementing asynchronous callout support 853
  callout configuration overview 849

Open Transaction Manager Access (OTMA) *(continued)*
callout requests 849
callout, asynchronous
correlating responses 853
capabilities 754
client
initiating protected transactions 836
client descriptors
DFSOTMA 777
limit 778
CM0 output
acknowledgment time out 847
timeout for acknowledgment 847
commit-then-send output
acknowledgment time out 847
timeout for acknowledgment 847
configuration parameters 767
configuring 767
conversational transactions
terminating 806
coupling facility
cross system 753
XCF 753
cross system coupling facility 753
D descriptors
limit 778
default client values, setting 777
descriptors
limit 778
maximum 778
descriptors, specifying 773
destination descriptors
limit 778
DFSOTMA descriptor 777
distributed security credential propagation 812
enabling 767
GRNAME 768
IBM MQ
synchronized tpipes, defining 772
IMS administration 784
IMS environments supported 754
IMS-to-IMS TCP/IP communications flow 759
installing 767
introduction 753
M descriptors
DFSOTMA 777
limit 778
member descriptors
DFSOTMA 777
limit 778
member name override, enabling 772
message
prefix length 791
message flow for IMS-to-IMS TCP/IP communications 759
message prefix
message-control fields, explanations 862
security data fields, explanations 884
user data fields, explanations 886
MSC (Multiple Systems Coupling)
processing remote transactions in an IMSplex 719
MULTIRTP
enabling 770
super member support, changing 771
Open Transaction Manager Access (OTMA) *(continued)*
MULTIRTP *(continued)*
super members 771
network security credential propagation 812
OTMA C/I
Timing out OTMA C/I sessions 892
OTMA client descriptor 773
OTMA client, introduction 821
OTMA destination descriptor
defining 776
masking destination names 777
OTMA= 768
OTMAASY 772
OTMAMD 772
OTMANM 768
OTMASE 769
OTMASP 772
parameters
GRNAME 768
OTMA= 768
OTMAASY 772
OTMAMD 772
OTMANM 768
OTMASE 769
OTMASP 772
PROCLIB member DFSPBxxx 767
program-to-program switches
asynchronous output, specifying 772
protocol
overview 756
response flags
delayed acknowledgement support 862
restrictions and requirements 791
save area prefix (SAP)
allocating 805
limiting 805
security
asynchronous hold queues 814
general considerations 815
levels of checking, defining 769
specifying OTMA client security 809
specifying OTMA member security 809
system definition 808
security, RACF 806
server definition 754
starting
defining when OTMA starts 768
Startup procedures 767
super members
MULTIRTP 771
MULTIRTP setting, changing 771
parallel RESUME TPIPE request, support for 771
parallel RESUME TPIPE support, changing 771
RESUME TPIPE requests, multiple active 771
synchronous callout support
MULTIRTP 858
parallel processing of RESUME TPIPE requests 858
system definition
security 808
time out CM0 acknowledgment 847
tpipes
MULTIRTP 769
MULTIRTP, enabling 770

Open Transaction Manager Access (OTMA) *(continued)*
tpipes *(continued)*
parallel processing of RESUME TPIPE requests, enabling support 770
parallel RESUME TPIPE request, support for 769
RESUME TPIPE requests, multiple active 769
transaction expiration
in seconds 804
overview 803
STCK format 804
XCF
group name, defining 768
member name, defining 768
operator logical paging
paged messages 552
paging errors 549
QXFR FM header 571
SLU P, MFS option 913
sync points 508
VTAM indicators 518
when in effect 514
OPNDST (/OPNDST) command
ISC TCP/IP sessions 601
relation to MODETBL keyword 418
OPNDST command
logging on 89
results of using 97
OPTACK option
Fast Path 919
Finance Communication System 947
SLU P 917, 944
options, CICS
system definition 613
table preparation 613
OPTIONS= DPAGE or PPAGE
MFS DPM 548, 939
output message 571
orderly session termination
CICS 621
ISC 499
VTAM 929
OTMA
/DISPLAY TRANSACTION
output format 895
ACEE flood control 800
ALTPCB output, delivering 789
application-data section of message prefix 886
architected transaction attributes 895
asynchronous hold queue 848
asynchronous output
IMS Connect 322
buffer pool
HIOP 793
LUMP 793
callout
IBM MQ 856
SOAP Gateway 855
callout requests
IMS Connect 322
client
definition 821
client-bid message flow 832
clients
routing 757
OTMA *(continued)*
CM0 output, managing 845
CM0 output, purging 845
CM0 output, super member 847
CM1 ACK timeout 803
CM1 transactions
timeout 803
commands
architected output 757
IMS 824
OTMA terminology 824
commit
summary of processing 825
commit-then-send message flow 834
commit-then-send output, managing 845
commit-then-send output, purging 845
commit-then-send output, rerouting 846
commit-then-send output, super member 847
conversations, administering IMS 784
DBCTL unsupported 792
DL/I calls
CHNG 816
INQY 816
PURG 816
SETO 816
express PCB and program switch 818
Front-End Switch, unsupported 792
I/O PCB, and program switch 818
IMS Connect
specifying OTMA ACEE aging value 199
timeout specifications 310
IMS conversations 784
IMS restart processing
IMS to IMS TCP/IP messages 791
tpipe WAIT_R status 791
IMS-to-IMS TCP/IP communications
format of output messages 780
IMS Connect support for OTMA 153
messages, format on output 780
overview of OTMA support 778
overview of super member support 779
reconnecting automatically 154
super member 153
transaction code, specifying 780
LTERM 762
message
IMS resynchronization support 837
queue data set size 796
recoverable 837
message encryption, restriction 792
message flood detection 796
message format
IMS-to-IMS TCP/IP communications 780
Message Format Service, unsupported 791
message prefix
cancel resume tpipe request format 882
chain flag (TMAMCCHN) 866
command type (TMAMCTYP) 864
commit-confirmation flag (TMAMCCCI) 863
message type (TMAMCMGT) 862
no messages on tpipe hold queue format 882
overview 822
prefix flag (TMAMCPFL) 867

OTMA *(continued)*
message prefix *(continued)*
processing flag (TMAMCPFG) 865
reason code (TMAMCRSC) 867
response flag (TMAMCRSI) 862
send-sequence number (TMAMCSSN) 867
sense code (TMAMCSNC) 867
sense code (TMAMCSNS) 867
TMAMAGNG (userid aging value) 867
TMAMCCCI (commit-confirmation flag) 863
TMAMCCHN (chain flag) 866
TMAMCMGT (message type) 862
TMAMCPFG (processing flag) 865
TMAMCPFL (prefix flag) 867
TMAMCRSC (reason code) 867
TMAMCRSI (response flag) 862
TMAMCSNC (sense code) 867
TMAMCSNS (sense code) 867
TMAMCSSN (send-sequence number) 867
TMAMCTNM (tpipe name) 866
TMAMCTYP (command type) 864
tpipe name (TMAMCTNM) 866
userid aging value (TMAMAGNG) 867
message prefix, application-data section 886
message prefix, security-data section 883
message prefix, user-data section 886
messages
client-bid flow 832
commit-then-send flow 834
displaying current workload 794
flow, client-bid message 832
flow, commit-then-send message 834
flow, server-available message 833
prefix, contents 832–834
sample flows 831
server-available flow 833
MSDB, read only access 792
performance
resource monitoring 799
prefix
contents 832–834
program switch
usage scenarios 817
protected conversation messages, restrictions 791
protocol commands
cancel resume tpipe request format 882
no messages on tpipe hold queue format 882
overview 799
server state 878
purging output
NAK responses 846
RACF security classes 812
recoverable messages 837
rerouting output
NAK responses 847
send-only output 846
resource management
transaction instance block (TIB) 795
resource monitoring 799
resource recovery services 791
resources
monitoring 799
restart processing 785
resynchronization
OTMA *(continued)*
resynchronization *(continued)*
assumptions 837
overview 837
sample messages 887
security
asynchronous hold queue 813
DFSYRTUX 814
IMS-to-IMS TCP/IP connections 815
message prefix security specifications 811
OTMA Resume TPIPE security exit routine 814
specifying 808
security-data section of message prefix 883
send-only
rerouting output 846
server state protocol command 878
server-available message flow 833
shared queues
ALTPCB output, retrieving 789
shutdown
client notification 791
overview 791
SLU 2 Transaction flow
standard 760
state data
REQresynch command format 872
state-data
resume output for all tpipes format 880
resume output for tpipe format 880
super member
IMS-to-IMS TCP/IP communications 153, 779
synchronous program switch 851
terminal control commands unsupported 792
termination
client notification 791
overview 791
timeout
CM1 transactions 803
timeout interval
specifying for OTMA client 802
timeout, ACK 801
TMAMAGNG 867
TMAMCCCI 863
TMAMCCHN 866
TMAMCMGT 862
TMAMCPFG 865
TMAMCPFL 867
TMAMCRSC 867
TMAMCRSI 862
TMAMCRSQ 867
TMAMCSEQ 868
TMAMCSNC 867
TMAMCSNS 867
TMAMCSSN 867
TMAMCTNM 866
TMAMCTYP 864
TMAMRTID 868
transaction attributes segment 895
transaction code, specifying
IMS-to-IMS TCP/IP communications 780
user-data section of message prefix 886

OTMA *(continued)*
XCF macros, reassembly 791
OTMA (Open Transaction Manager Access)
/START OTMA 768
ACEE, refreshing 810
administering 767
asynchronous hold queues
security 814
benefits 755
callout
implementing asynchronous callout support 853
callout configuration overview 849
callout requests 849
callout, asynchronous
correlating responses 853
capabilities 754
client
in XRF environment 785
initiating protected transactions 836
client descriptors
DFSOTMA 777
limit 778
CM0 output
acknowledgment time out 847
timeout for acknowledgment 847
commit-then-send output
acknowledgment time out 847
timeout for acknowledgment 847
configuration parameters 767
configuring 767
conversational transactions
terminating 806
coupling facility
cross system 753
XCF 753
cross system coupling facility 753
D descriptors
limit 778
default client values, setting 777
descriptors
limit 778
maximum 778
descriptors, specifying 773
destination descriptors
limit 778
DFSOTMA descriptor 777
distributed security credential propagation 812
enabling 767
GRNAME 768
IBM MQ
synchronized tpipes, defining 772
IMS administration 784
IMS application programs 754
IMS environments supported 754
IMS-to-IMS TCP/IP communications flow 759
installing 767
introduction 753
IOPCB 757
M descriptors
DFSOTMA 777
limit 778
member descriptors
DFSOTMA 777
limit 778
OTMA (Open Transaction Manager Access) *(continued)*
member name override, enabling 772
message
prefix length 791
message flow for IMS-to-IMS TCP/IP communications 759
message prefix
message-control fields, explanations 862
security data fields, explanations 884
user data fields, explanations 886
message switch 817
message-control information section of message prefix 859
MSC (Multiple Systems Coupling)
processing remote transactions in an IMSplex 719
MULTIRTP
enabling 770
super member support, changing 771
super members 771
network security credential propagation 812
OIM TCB 805
OTMA C/I
Timing out OTMA C/I sessions 892
OTMA client descriptor 773
OTMA client, introduction 821
OTMA destination descriptor
defining 776
masking destination names 777
OTMA= 768
OTMAASY 772
OTMAMD 772
OTMANM 768
OTMASE 769
OTMASP 772
parameters
GRNAME 768
OTMA= 768
OTMAASY 772
OTMAMD 772
OTMANM 768
OTMASE 769
OTMASP 772
Prerouting exit routine (DFSYPRX0) 757
PROCLIB member DFSPBxxx 767
program switch 817
program-to-program switches
asynchronous output, specifying 772
protected messages 820
protocol
overview 756
response flags
delayed acknowledgement support 862
restrictions and requirements 791
resynchronization protocol 840
save area prefix (SAP)
allocating 805
limiting 805
security
general considerations 815
levels of checking, defining 769
specifying OTMA client security 809
specifying OTMA member security 809
security, RACF 806
server definition 754

OTMA (Open Transaction Manager Access) *(continued)*
starting
defining when OTMA starts 768
Startup procedures 767
super members
MULTIRTP 771
MULTIRTP setting, changing 771
parallel RESUME TPIPE request, support for 771
parallel RESUME TPIPE support, changing 771
RESUME TPIPE requests, multiple active 771
synchronous callout requests
MULTIRTP 858
parallel processing of RESUME TPIPE requests 858
system definition
security 808
time out CM0 acknowledgment 847
tpipes
MULTIRTP 769
MULTIRTP, enabling 770
parallel processing of RESUME TPIPE requests, enabling support 770
parallel RESUME TPIPE request, support for 769
RESUME TPIPE requests, multiple active 769
transaction expiration
in seconds 804
overview 803
STCK format 804
XCF
group name, defining 768
member name, defining 768
OTMA administration
target members 792
tmembers 792
OTMA C/I
Timing out OTMA C/I sessions 892
OTMA client
naming conventions for 822
OTMA client descriptors 773
OTMA descriptors
limit 778
maximum 778
OTMA destination descriptors 774
OTMA Destination Resolution exit routine (DFSYPRX0)
IMSplexes 720
MSC 720
multiple IMS systems 720
OTMA Resume Tpipe Security exit routine (DFSYRTUX) 781
OTMA= parameter 768
OTMAASY option 819
OTMAASY parameter 772
OTMAMD parameter 767, 772
OTMANM parameter 767, 768
OTMASE parameter 767, 769
OTMASP parameter 767, 772
output
available at ISC restart 494
controlling 439
MFS DPM 939
output algorithms
RU chain 561
VLVB 561
output bracketing 942
output component
defining 467
output component *(continued)*
relationship to input component 462
selection
identification number 464, 907
modifying 463, 907
system messages 545, 907
output component ID byte, output message 940
output component protection, extended 914
output devices, control characters by type 438
output editing option, SLU P 916
output errors, MFS online detection 548
output function management headers, ISC 629
output message
between brackets
design considerations 911
figure 910
how handled 910
message switching 947
description 936
Fast Path
Finance systems 920
SLU P systems 920
Finance Communication System
multiple transmission 936
read type field (SMSCRT) 936
read-flags field (SMSCRF/SMSCRE) 936
formatting, activating MFS 547, 943
ISC bracketing 518
MFS DPM 548
null
purpose 936
when sent 942
output bracketing 942
output FM header
Finance format 939
ID byte 940
message descriptor byte (Finance) 940
message descriptor byte (SLU P) 941
MFS data bytes (Finance) 940
MFS data bytes (SLU P) 942
response requests 944
SLU P format 940
segmenting 936
SLU P system
multiple transmission 936
read type field (SMSCRT) 936
sync point requested, ISC 508
temporarily stopping 929
types 936
when committed 508
output message structure 360
output protocols, determining 467
output response requested by message type 945
overload avoidance, MSC 686

P

page advance function (NEXTPP), MFS 913
page delete function, MFS 548
page protection state 396
paging errors
ISC 531
online detection, MFS 549

parallel sessions
bind requirements 490
ISC, defining IMS-to-IMS sessions 478
users permitted 463
parameter
OTMAMD 767
OTMANM 767
OTMASE 767
OTMASP 767
parameters
OTMA configuration parameters, summary 767
partner systems in MSC (Multiple Systems Coupling) linking 669
PassTicket
IMS Connect 193
replay protection 198
passwords
IMS Connect mixed-case passwords 192
IMS Connect, changing RACF password phrases 191
IMS Connect, changing RACF passwords 190
PCB (program communication block)
alternate PCB 464
for an LTERM not in I/O PCB 907
I/O PCB 464
PDSE
external subsystem module table 121
performance 792
Persistent Session Tracking
Termination of 411
persistent socket connections 300
physical links
defining
buffer sizes 695
CTC 696
MTM 696
TCP/IP 697
VTAM 697
MSC (Multiple Systems Coupling) 667
types 694
physical terminals
defining 70
device class control 438
separating input and output devices 392
ping
IMS Connect availability 342
PING response
format 244
PL/I
XML conversion support
overview 154
pool manager
MFS description 433
MFSTEST 433
prefix
of OTMA message 822
rules 823
syntax 859
Prerouting exit routine (DFSYPRX0)
basic message flow 757
customizing IMS for OTMA 781
OTMA usage restrictions 791
primary error recovery procedure 902
primary resource name (PRN) 562
printed output
/ASSIGN command 439
3270 printer components 440
3270R and ETO 440
ASSIGN (/ASSIGN) command 439
candidate printers 440
controlling 439, 441
operational considerations 441
printer component 439
shared printers 441
spooled output control 439
VTAMLIST definitions 440
printers
candidate 440
sharing 441
private context 817
PRN (primary resource name)
ATTACH FM header 562
message routing, ISC 469
SCHEDULER FM header 562
processing affinity
in an MSC-IMSplex configuration 715
processor utilization, MSC 686
PROFILE security level 807
program communication block (PCB) 464
program specification block (PSB)
CICS DRA
coding guidelines 743
program switch
CM0 messages 817
CM1 messages 817
race condition 819
synchronous 851
usage scenarios
for protected transactions 820
single-stream 818
to multiple programs 819
with express PCB 818
with OTMAASY option 819
without ISRT to I/O PCB 818
program-to-program (P2P) 817
program-to-program switch
conversational transactions 684
destination name 680
nonconversational transactions 684
remote destination verification 688
protected
conversations 817
transactions 836
protected resource, definition 27
protection
display screen 914
extended output component 914
protocol
output 467
restrictions on IMS-to-IMS sessions 480
protocol level 284
protocols
IMS Connect conversational
send-then-commit, sync level=confirm, ACK response 289

protocols *(continued)*
IMS Connect conversational *(continued)*
send-then-commit, sync level=confirm, NAK response 290
send-then-commit, sync level=none, client terminated transaction 288
send-then-commit, sync level=none, program terminated transaction 287
IMS Connect Send Only 296
OTMA
server state protocol command 799
PSB (program specification block)
CICS DRA
coding guidelines 743
pseudoabend U0830
avoiding 725
PSTOP state 422
PURGE
error recovery procedure
begins 525
DFC protocol 525
DFC support layer 525
ends 525
multichain 525
single 525
spanning chains 525
state after selective receiver purge 527
state after sender purge 526
sender ERP 536
purge function, IMS Connect 291
PURGE state 422

Q

QCF 786
QEC command 929
QERROR state 396
QGET FM header
description 553
format 567
QGETN FM header
description 553
format 568
QLOCK state 396
QMODEL FM headers
CICS 634
formats 567
QGET 553
QGETN 553
QPURGE 554
QSTATUS 554
QXFR 554
reply or output 554
request or input 553
supported by IMS 552, 567
QPURGE FM header
format 569
received by IMS 554
QSTATUS FM header
format 570
sent by IMS 554
QSTOP state 422
qualifying an LU name 52
queue control facility
queue control facility *(continued)*
identifying message categories 786
support for non-shared queues 785
support for shared queues 785
queue empty indication 529
queue rotation 520
queued subsystem 622
queues
balancing group 380
logical terminals 392
shared 373
QUIESCE (/QUIESCE)
command
ISC TCP/IP sessions 603
quiesce-at-end-of-chain command, VTAM 929
QXFR FM header
as output 554
format 571

R

race condition
avoiding 819
defined 819
race, BIND 491
RACF
ACEE
refreshing OTMA ACEEs 810
command security 402
IMS Connect mixed-case passwords 192
IMS Connect support for 175
IMS Connect support for PassTicket 193
IMS Connect user ID cache 178
IMS Connect, changing password phrases 191
IMS Connect, changing passwords 190
IMS Connect, default user ID 178
IMS Connect, enabling direct support 176
IMS Connect, enabling statistics 177
IMS Connect, generic return code or message 176
mixed-case passwords, IMS Connect 192
OTMA security classes 812
PassTicket and IMS Connect 193
password phrases, changing IMS Connect 191
passwords, changing IMS Connect 190
RACF (Resource Access Control Facility)
APPC transaction security 57
ETO security 370
FACILITY class definition 768
OTMA restrictions 792
OTMA security 806
security for MSC (Multiple Systems Coupling) 705
unauthorized terminal use 401
RACF PassTicket
IMS Connect
DRDA clients 194
RACF PassTicket for IMS Connect Client connections to IMS OM 197
RACROUTE call
IMS Connect
error responses on calls from sample exits 185
RAP FM header
description 544
example 544
format 573

RAP FM header *(continued)*
MFS 555
Rapid Network Reconnect
and IMS Shutdown 411
Changing Levels of Support 410
defining level of persistent support for VTAM 423
defining level RNR support for VTAM 423
Defining VTAM for 423
Persistent Session Tracking 410
Signon Security 412
Specifying Levels of Support 410
Terminal Reconnect Protocols 411
using with XRF or VGR 411
Rapid Network Reconnect (RNR)
planning 409
RCVYCONV= 398
RCVYFP= 398
RCVYRESP= 398
RCVYSTSN= 398
RDPN (Return Destination Process Name)
ATTACH FM header 563
message routing, ISC 470
SCHEDULER FM header 563
read flags field, output messages 936
read type field, output messages 936
ready-to-receive (RTR) command 531
receive-any buffers 417
recoverable
resources 836
transactions 838
recoverable input, acknowledging 917
recoverable message sent, sequence numbers 925
recoverable messages, ISC sync points 508
recoverable versus irrecoverable messages 943, 945
recoverable versus irrecoverable transactions 47
recoverable-inquiry transactions, response requirements
IMS 944, 946
ISC 508, 510
recovering send-then-commit 825
recovering transactions in APPC 47
recovery
Fast Path 399
resource status 397
recovery coordinator 113
recovery environment
distributed resources, definition 29
local resources, definition 29
recovery of OTMA messages, selective 785
recovery procedures 407
recovery token 118, 135
RELQ command 929
remote control of IMS 479
remote destination name 680
remote destination verification
MSC (Multiple Systems Coupling) conversations 688
system integrity 684
remote LTERMs 681
remote system
MSC (Multiple Systems Coupling)
defined 671
remote terminal operator (RTO) 418
reply resynch 840
reports
IMS Monitor 711
reports *(continued)*
MSC 711
REPresynch 840
REPresynch command 872
REQresynch 840
REQresynch command 872
request resynch 840
reroute function, IMS Connect 293
reset attached process FM header 544
resource
MSC (Multiple Systems Coupling) considerations 685
status classification 397
status recovery 397
status recovery mode 397
Resource Access Control Facility (RACF) 705
resource coordination 118
resource manager
IMS 30
Resource Manager
IMS TM resources, benefits of RM 378
Resource Manager (RM)
IMS TM resources
managing 378
resource manager, definition 27
resource name uniqueness
disabling enforcement 379
resource recovery services
OTMA restrictions and requirements 791
protected conversation messages, OTMA restrictions 791
resource structure
benefits 378
IMS TM resources
managing 378
resource translation table 114
resource type consistency
disabling enforcement 380
resources
TM
sharing 379
response mode
dynamic establishment 463
errors
ISC 505
ISC
input 627
terminal
definition 908
design considerations 589, 590, 909
effects of specifying transaction-dependent 908
figure 908
introduction 394
methods of termination 910
restrictions 909
sizes of message queue data sets 937
sizes of workstation output buffers 937
station-by-station 908
when activated 908
response requests, output message 945
response requirements
IMS commands and indicators 929, 943
inquiry transactions
irrecoverable 506, 946
recoverable 506, 946

response requirements *(continued)*
irrecoverable-inquiry transactions 421
ISC messages 508
LU 6.2 application program 421
MFS control requests 947
output ISC messages 508
verifying IMS receipt 508, 947
VTAM indicators and commands 456, 929
response time 432, 712
restart
coding CICS applications 640
output available, ISC 494
restart and recovery 635
restart processing
OTMA
messages rerouted to timeout queue 791
restart transaction, CICS 611
RESUME TPIPE
alternate client ID 335
IRM_RT_ALTCID 335
retrieving output for another client 335
RESUME TPIPE protocol
example flows 324
RESUME TPIPE request
alternate client ID 329
IMS Connect
parallel processing 327
parallel processing, diagnosing problems 329
parallel processing, enabling 328
parallel processing, implementing 329
parallel processing
alternate client ID 329
diagnosing problems 329
implementing 329
IMS Connect 327
IMS Connect, enabling from 328
retrieving asynchronous output
IMS Connect 322
RESUME TPIPE requests
parallel processing
callout messages 858
resynchronization
deferred 841, 842
flow 841
OTMA protocol 840
OTMA, overview 837
sample message 844
resynchronization, message 491
Return Destination Process Name 563
Return Primary Resource Name 563
Returned Client ID
format 242
RM affinity 398
RMM (request mod message)
format 242
RNR 705
RNR (Rapid Network Reconnect)
planning 409
routing
messages
in an MSC-IMSplex configuration
713
routing code, definition 380
routing exit routines
routing exit routines *(continued)*
link routing 732
MSC (Multiple Systems Coupling) conversations 688
program routing 732
Routing exit routines
Terminal Routing exit routine 731
routing messages
destination name 680
ISC examples 471
parameters 470
SYSIDs 680
routing path
MSC (Multiple Systems Coupling) 673
RPRN (Return Primary Resource Name)
ATTACH FM header 563
message routing, ISC 470
SCHEDULER FM header 563
RQ* messages, LUSTATUS command 528
RQD* 455, 508, 512
RQE* 455, 508, 512
RQR command, SLU P 904
RR_BACKED_OUT 26
RR_OK 26
RR_PROGRAM_STATE_CHECK 26
RRS 836
RRS (z/OS Resource Recovery
Services)
description 27
resource recovery with 27
RRSAF 111
RSHUT command 531
RSM (request status message)
format 243
RTO (remote terminal operator) 418
RTR (ready-to-receive) command
Fast Path 950
IMS functions 950
protocol 531
resetting component status to unprotected 915
summary 531, 950
RTT (resource translation table) 114
RU (request unit) chaining in ISC 524

S

sample
message
client-bid 887
response 887
transaction 887
OTMA message 887
sample programs
using ISC between IMS and CICS 657
save area prefix (SAP)
OTMA input messages 805
SBI (stop bracket initiation) command
CICS 621
IMS-CICS session 621
session shutdown 540
SC (session control) protocols
BIND parameters 496
binding sessions
negotiable versus nonnegotiable BIND 490
parallel session 490

SC (session control) protocols *(continued)*
binding sessions *(continued)*
resolving a race 491
single session 490
synchronizing sessions 491
message resynchronization
commands used 491
designing procedures 492
when required 491
session initiation
completing 499
ISC VTAM 488
session states 498
session termination
abnormal 499
normal 499
STSN flow
primary-to-secondary half session 500
secondary-to-primary half session 502
STSN format 503
XRF complex, establishing an ISC session 489
XRF complex, establishing connection 489
XRF with MNPS, establishing connection 489
XRF with USERVAR, establishing connection 489
SCA (system control area), ISC support 468
SCHEDULER FM header
ATTACH 545
chained message support 545
example 577
format 573
IMS-CICS session 632
introduction 544
MFS 550
parameter description 561, 563
SCHEDULER FM headers
request for asynchronous execution 461
scheduler message block (SMB)
OTMA 757
scheduling
algorithm 388
Fast Path messages 380
scratchpad area (SPA) 389
screen protection, Finance Communication System
BID option 914
screens
protection 396
unprotected screen option 436
SDF II
MFS formats 433
SDT (start data traffic) command
completing session initiation 499
IMS 928
secondary logical unit
design considerations 458
first speaker in ISC 458
secondary logical unit type P 908
security
APPC transactions 57
distributed security credential propagation 179
DRDA clients 194
IMS Connect
changing password phrases 191
changing passwords 190
default RACF user ID 178
security *(continued)*
IMS Connect *(continued)*
exit routine 183
mixed-case passwords 192
overview 156
PassTicket 193
RACF, enabling direct support 176
RACF, enabling statistics 177
RACF, generic return code or message 176
specifying OTMA ACEE aging value 199
trusted users 198
IMS Connect to IMS Connect connections 181
IMS-to-IMS TCP/IP connections 181
intermediate IMS
MSC 705
ISC
ISC TCP/IP connections 595
MSC
intermediate IMS 705
MSC (Multiple Systems Coupling) 705
network security credential propagation 179
OTMA
specifying individual member security 809
OTMA ACEE aging value
IMS Connect 199
OTMA clients, overview 754
OTMA RACF security levels 806
OTMA security-data section of message prefix 883
PassTicket
IMS Connect 193
password phrases
changing, IMS Connect 191
passwords
changing, IMS Connect 190
IMS Connect mixed-case passwords 192
RACF PassTicket 194
trusted users
IMS Connect 198
Security
IMS Connect
errors on RACROUTE calls from sample exits 185
security considerations
CICS 627
security levels
CHECK 807
FULL 807
NONE 806
PROFILE 807
security options
APPC/IMS, RACF 403
APPC/IMS, SAF 403
command authorization
DFSCCMD0 Command Authorization exit routine 403
user ID 402
ETO
user ID 404
password
user ID 403
RACF 401
security profile
DFSCTRN0 Transaction Authorization exit routine 402
user ID 402

security options *(continued)*
signon verification 401
transaction authorization
RACF 402
transaction command 403
transaction security
RACF 57
UACC (NONE) 57
security support
PassTicket replay protection 198
security-data section of OTMA message prefix 883
selective receiver ERP
description 531
sense codes 532
send and receive protocol
bracketing
input 936
output 942
IMS 932
SEND INVITE EXEC command, CICS 605
SEND LAST EXEC command, CICS 607
send-only protocol 296
Send-only protocol
with acknowledgment 297
with error 298
with serial delivery 298
send-only transactions
rerouting output 293
send-then-commit
ACK/NAK timeout value 283
flow 829
with Confirm flow 830
send-then-commit transactions (CM1)
ACK timeout 803
send/receive and bracket protocol 932
SEND/RECEIVE EXEC command, CICS 605
sender ERP, sense codes 537
sending IMS commands from CICS 626
sense code
definition 538
error 538
received during ISC 538
selective receiver ERP 531
sender ERP 537, 538
sent during ISC 539
separating input and output devices 392
sequence numbers
definition 822
description 496
maintaining 493
management 503, 925
recoverable 823
send-sequence numbers 823
storage 503
use 496, 925, 926
sequential buffering
CICS 744
serial transactions
in an MSC network 701
server resynch 840
server state protocol command
overview 799
server-available exchange
OTMA message flow 833
server-available exchange *(continued)*
OTMA message prefix contents 833
servers
OTMA definition 754
services
ATRABCK 836
ATRACMT 836
ATREINT 836
CRGGRM 836
CRGSEIF 836
CTXBEGC 836
CTXEINT 836
CTXSWCH 836
services available to ESAP 119
session
binding 490
definition 367
establishing 372
ISC VTAM, establishing 488
parallel 490
remote control 479
requirements to bind ISC 490
single 490
session initiation
bind parameters
Finance Communication System 924
IMS-CICS 620
ISC VTAM 488
possible session states 498
requested by
CICS 620
master terminal operator, IMS 488, 923
network operator, z/OS VTAM 488, 923
system definition 488, 923
workstation 923, 936
step-by-step explanation 924
transmission sequence 924
ways to request 420
session local flag, ATTACH FM header 564
session parameters
establishing connection 488, 904
use 488, 904
session termination
abnormal 499, 621
BID option specified, effects 949
CICS 621
conditional versus unconditional 458
definition 499, 929
immediate
definition 621, 929
master terminal operator, IMS 499, 929
network operator, z/OS VTAM 499, 929
station 499
workstation 929
normal 499, 621
orderly
definition 930
initiated 540, 930
requested by CICS 621
sequence 540, 930
summary (figure) 929
symmetrical (SBI/BIS) in ISC 540
SET IPCONN CICS command
ISC TCP/IP sessions 602

set-and-test-sequence-numbers (STSN) command 500
setting inquiry parameters 118
shared queues
  ALTPCB output and IMS Connect 174
  APPC and OTMA messages
    processing MSC remote transactions 719
  benefits 374
  commit-then-send messages 787
  definition 373
  EMH queue option
    overview 380
  enabling OTMA 783
  environment
    components of, illustration 375
    operating in, overview 373
    required components of 374
  IMS Connect and ALTPCB output 174
  link definitions in an IMSplex
    deleting 719
  message routing
    in an MSC-IMSplex configuration 713
  MSC (Multiple Systems Coupling)
    coexistence 713
    sharing MSNAME definitions and SYSIDs 716
  MSC MSNAME definitions
    duplication 718
    sharing MSNAME statements and SYSIDs 716
  MSNAME definitions
    deleting 718
  OTMA
    ALTPCB output, retrieving 789
  output message queue count 786
  program-to-program message switch 820
  program-to-program switch 787
  pseudoabend U0830
    avoiding 725
  send-then-commit messages 787
  SYSIDs
    cloning MSC SYSIDs in an IMSplex 719
    managing MSC SYSIDs in an IMSplex 719
    sharing MSNAME definitions and SYSIDs 716
    when an IMSplex and MSC network coexist 719
  tpipe status 789
  unsolicited messages 787
  z/OS system log, role of 375
sharing printers
  between systems 441
shutdown
  OTMA
    client notification 791
    overview 791
shutting down an IMS network 487, 931
SIGNAL command
  protocol 540
  VTAM 951
signing off 372
signing off, definition 372
signing on, definition 372
signon and static terminals 417
Signon exit routine (DFSSGNX0)
  ETO
    associated printing 96
signon process for external subsystem connections 116
single session, ISC bind requirements (IMS-to-IMS) 478
single-stream program switch 818
SINGLE1 parameter 467, 519
SINGLE2 parameter 467, 519
SLU 2 Transaction flow
  with OTMA 760
SLU P
  application program
    converting from Finance 901
    functions available 900
    MFS considerations 900
    XRF considerations 901
  bracket and send/receive protocols 905, 932
  controller-detected errors 948
  facilities 907
  IMS-detected errors 948
  input component definition 908
  message resynchronization
    controller application program 924
  MFS DPM option 939
  network
    major parts 899
  sample configuration 900
  SCAN/NOSCAN 901
  session termination
    immediate 930
  terminals supported 899
  VTAM
    commands 947
    commands and indicators 902
    facilities 902
    indicators 947
  workstations 900
  XRF 919
  XRF complex, establishing connections 905
SNA commands 456
SNA QUIESCE 396
socket connections
  IMS Connect
    IMS-to-IMS TCP/IP communications 302
    setting socket type for IMS TM clients 301
  IMS-to-IMS TCP/IP
    communications
  cleanup 302
  persistence 302
  termination scenarios 302
  non-persistent 301
  persistent 300, 340
  transaction 301
  types 300
sockets
  IMS Connect
    processing for transactions 303
  IMS Connect maximum and UNIX System Services 305
  maximum number for IMS Connect 304
  reserving 306
  reset threshold 307
  setting percentages for warnings 307
  UNIX System Services socket limits 305
  warning threshold 307
SPA (scratchpad area)
  characteristics 389
  MSC (Multiple Systems Coupling) conversations 687
  size 684, 687

SPA (scratchpad area) *(continued)*
transaction code field 390, 395
specifying external subsystems to IMS 114
splitting databases, MSC (Multiple Systems Coupling) 686
SPOOL command
output control 439
spooled output control 439
SRVresynch 840
SRVresynch command 872
START (/START) command
starting a SLU P network 923
start data traffic (SDT) command 499
start lists, VTAM 419
START/RETRIEVE EXEC commands, CICS 610
starting a SLU P network 420
starting an IMS network
making IMS ready
/START command 487, 923
results 487, 923
prerequisites 420, 923
starting workstations 420
state-data
format
client-bid commands 868
for resume output for hold queue for tpipe 881
REPresynch commands 872
server-available commands 868
SRVresynch commands 872
TBresynch commands 873
transaction-related information 874
section of message prefix 868
station, user
definition of as logical unit 899
device selection 899
Statistical Analysis Utility and MSC 710
statistics
bandwidth mode, capacity 707
logical links, benchmarking 706
logical links, buffer sizes 707
logical links, high value statistics 710
logical links, link capacity 707
logical links, MSC 705
logical links, resetting statistics 706
MSC link types, optimum 706
MSC links
buffer size and format in bandwidth mode 708
buffer size and format in non-bandwidth mode 709
buffers, determining optimum size 707
MSC logical links 705
MSC logical links, benchmarking 706
MSC logical links, buffer sizes 707
MSC logical links, high value statistics 710
MSC logical links, link capacity 707
MSC logical links, resetting statistics 706
status
nonrecoverable 397
recoverable 397
resource
classification 397
significant
command 397
end-user 397
status recovery mode
status recovery mode *(continued)*
GLOBAL 398
LOCAL 398
NONE 398
RCVYCONV= 398
RCVYFP= 398
RCVYRESP= 398
RCVYSTSN= 398
resource types 398
STEPLIB
TM and MSC Message Routing and Control user exit routine 730
STOP (/STOP) command 396
stop bracket initiation (SBI) command
IMS-CICS session 621
stop bracket initiation command 540
STOP state 396, 422
stopped status 137
stopping
workstations 499
structure
creation and deletion 65
list
definition 374
overflow, definition 375
primary, definition 375
pair, definition 374
structure recovery data set (SRDS)
overview 375
STSN command
action code 926
action code format 503, 926
action required 926
controller sequence number, verification 926
Finance Communication Systems 926
flow 500
message resynchronization 926, 928
response
requirements 927, 928
response requirements 926
summary 928
sync points 496
VTAM sequence number, verification 926
STSN function, requirements for ETO 105
STSN support for ETO devices 106
SUBPOOL macro 463, 483
subpools users
dynamic allocation 463
VTAM 463
subsystem
direct-control 622
external, definition 370
queued 622
subsystem connections 116
Subsystem Startup Service
using 132
subsystem termination 136
super member
IMS Connect 153
IMS Connect support 336
IMS-to-IMS TCP/IP communications 153, 779
managing CM0 output 847
OTMA 847

super members
MULTIRTP 771
parallel RESUME TPIPE requests
enabling 771
parallel processing 771
RESUME TPIPE requests
multiple active, support for 771
parallel processing 771
suspending output from IMS 929
switch_context service. 836
sync point
definition 388
input messages, ISC 507
ISC
CICS 627
input 510
output 512
requirements 508
Synch Level
CONFIRM 284
NONE 283
SYNCH 284
synchronization point
definition 388
synchronization, ISC half sessions 506
synchronous callout requests
MULTIRTP 858
OTMA configuration overview 849
OTMA support 850
OTMA support overview 849
parallel processing of RESUME TPIPE requests 858
synchronous processing
ATTACH FM headers, ISC 543
CICS 605
definition 461
synchronous program switch 851
syntax diagram
how to read xxiii
syntax, message prefix 859
SYS1.VTAMLST
ATCCONxx member 419
ATCSTRyy member 419
defining IMS as an application node 416
VTAM nodes 416
SYSERROR FM header
format 575
SYSID (system identifier 677
SYSID keyword
logical link paths 670
SYSID tables
deleting MSNAME definitions in an IMSplex 718
SYSIDs
cloning in an IMSplex 719
managing in an IMSplex 719
sharing among systems in an IMSplex 716
SYSMSG FM header
format 575
SYSMSG FM headers
ATTACH FM header 545
sending 545
types 546
Sysplex Distributor
IMS Connect 847
IMS Connect support for 155
Sysplex Distributor *(continued)*
OTMA support for 847
sysplex environment
definition 373
shared queues in 373
SYSSTAT FM header
format 575
SYSSTAT FM header, format 575
system
definition
IMS ISC sample 480, 485
messages, length 546
system control area (SCA) 468
system definition
macros
MSC macros 689
MSC
general considerations 689
implications 702
partner 694
verifying 703
MSC (Multiple Systems Coupling)
exit routines 701
local 693
macros 693
setting link priorities 699
system identification for logical link paths 670
system identifier (SYSID) 677
system log
MSC 711
system log, z/OS
shared queues and 375
system resources 792

T

TBresynch 840
TBresynch command 873
TCP/IP
calling out to external services from IMS 853
CICS
functions supported, ISC 589
ISC sessions, functions supported 587, 589
generic resources
affinity management for MSC 726
affinity persistence for MSC 728
affinity, clearing for MSC links 728
affinity, clearing in IMS Connect 729
IMS Connect support for MSC 152
MSC affinity management 726
MSC affinity persistence 728
MSC affinity, clearing 728
MSC affinity, clearing in IMS Connect 729
MSC and XRF 729
XRF and MSC 729
IMS Connect 139
IMS Connect message structures 213
IMS Connect support for RACF PassTicket 193
IMS Connect trusted users 198
IMS Connect-related TCP/IP settings 361
IMS Connect, client communications 142
IMS-to-IMS TCP/IP communications
IMS Connect support for MSC 151
IMS Connect support for OTMA 153

TCP/IP *(continued)*
IMS-to-IMS TCP/IP communications *(continued)*
overview of IMS Connect support 150
reconnecting automatically 154
super member support 153
ISC
CICS front-end transaction types 604
CICS sessions, functions supported 587, 589
CICS, functions supported 589
functions supported for CICS 589
IMS Connect, overview of support 148
ISC and ETO 61
ISC session restart 604
ISC session termination
abnormal 604
orderly 603
unconditional 604
ISC support
defining the link to CICS 599
defining the link to IMS Connect 598
dynamic terminal definition 596
sessions, starting 601
sessions, starting from CICS 602
static terminal definition 596, 597
terminal definition 596
ISC, support for
falling back to VTAM 598
overview 593
requirements 594
restrictions 595
security 595
switching from TCP/IP to VTAM 598
KeepAlive intervals for IMS Connect 309
message formats 213
MSC
affinity management with generic resources 726
affinity persistence in a generic resource group 728
affinity, clearing 728
affinity, clearing in IMS Connect 729
generic resources 726
generic resources and XRF 729
XRF and generic resources 729
MSC physical link
defining 697
Multiple Systems Coupling (MSC)
affinity management with generic resources 726
affinity persistence in a generic resource group 728
affinity, clearing 728
affinity, clearing in IMS Connect 729
generic resources 726
generic resources and XRF 729
XRF and generic resources 729
purge function for output messages 291
reroute function for output messages 293
restarting ISC sessions 604
security
connections between IMS systems 181
connections between instances of IMS Connect 181
specifying OTMA ACEE aging value 199
socket connections
non-persistent 301
persistent 300
transaction 301
terminating ISC sessions
TCP/IP *(continued)*
terminating ISC sessions *(continued)*
orderly 603
unconditionally 604
terminal
physical, defining 70
terminal control commands
OTMA unsupported 792
TERMINAL macro
defining an ISC session 483
MODETBL= keyword 418
terminal modes
conversation mode 394
exclusive 395
lock mode 395
response mode 394
SNA QUIESCE 396
Terminal Reconnect Protocols 411
Terminal Routing exit routine 731
terminals
attached through VTAM 383
Class 1 with XRF 385
Class 2 with XRF 385
COMPINOP state 396
component protection state 396
connections 384
conversation mode 394
definition 366
device class control 438
documenting requirements 383
ETO and exclusive mode 395
exclusive mode 395
INOP state 396
lock mode 395
logical
chains 391
master terminal 393
queues 392
relationship to physical terminals 390
LU 6.2 terminals and Fast Path 404
modes and states 394, 396
nonswitched communications network 393
NTO 394
operating modes 394
page protection state 396
profiles 415, 417
QERROR state 396
QLOCK state 396
response mode 394
screen protection state 396
separating input and output devices 392
small buffer devices 438
SNA QUIESCE 396
states 396
STOP state 396
support for, IMS 383
sysplex, in a
recovery status 400
test mode 395
terminating communications, MSC conversations 689
terminating connections with external subsystems 117
terminating ISC Extension conversations 506
termination
OTMA

termination *(continued)*
OTMA *(continued)*
client notification 791
overview 791
termination ECB 117, 136
termination requested by external subsystem 136
termination, session 499
test mode 395, 460
thread, external subsystem 116
time settings
IMS Connect
IMS DB clients 310
IMS TM clients 310
IMS-to-IMS TCP/IP connections 320
overview 310
IMS Connect timeout intervals 311
timeout
IMS Connect
IMS DB clients 310
IMS TM clients 310
IMS-to-IMS TCP/IP connections 320
input messages 311
overview 310
OTMA ACK timeout 801
timer settings 335
TM and MSC Message Routing and Control user exit routine
affinity routing 732
IMSplex affinity routing 732
JOBLIB 730
LINKLIST 730
message routing 730
STEPLIB 730
TM resource
resource name uniqueness
disabling enforcement 379
TM resources
sharing 379
sharing, disabling 379
TMAMAGNG 867
TMAMALTB 789
TMAMCCCI 863
TMAMCCHN 866
TMAMCMGT 862
TMAMCPFG 865
TMAMCPFL 867
TMAMCRSC 867
TMAMCRSI 862
TMAMCRSQ 867
TMAMCSEQ 868
TMAMCSNC 867
TMAMCSNS 867
TMAMCSSN 867
TMAMCTNM 866
TMAMCTYP 864
TMAMRTID 868
tmember operand 785
tpipe 762
tpipe operand 785
tpipe_Bid resynch 840
tpipes
idle, automatic removal 800
MULTIRTP
enabling 770
removal of idle tpipes 800
resource impact 797
RESUME TPIPE requests
multiple active, enabling support 770
multiple active, support for 769
parallel processing 769
parallel processing, enabling 770
retrieving asynchronous output for alternate client IDs 335
retrieving output
IMS Connect 322
WAIT_R status 791
trademarks 953, 954
traffic between two IMS systems 463
TRANSACT macro
EDIT=ULC 437
PRTY= keyword 699
translation to uppercase 434
transaction
code, definition 380
codes, unique 387
MSC statistics 712
multiple systems 667
states 422
Transaction Analysis utility 710
transaction code (remote destination) 680
Transaction Manager
introduction 365
Transaction Manager services 369
transaction pipe
and message flow 763
definition 757, 762
differences from LTERMs 763
differences from UNIX pipes 763
flow in full-duplex environment 764
in an OTMA client/server environment 763
naming conventions for 822
non-synchronized 762
number a client can create 759
removal of idle transaction pipes 762
synchronized 762
use of queues and message flow 763
using 762
transaction pipes
idle, automatic removal 800
removal of idle tpipes 800
resource impact 797
transaction socket connections 301
transaction sync point relationships 507, 627
transaction types
commands 947
definitions 421, 945
inquiry
definition 944
recoverable or irrecoverable 945, 946
ISC session, during 460
message switches
IMS 947
ISC 559
ISC examples 471, 474
supported by ISC, list 448

transaction types *(continued)*
test mode, in ISC 460
update 945
transactions
abends 639
commit-then-send 824, 826
conversational
terminating in OTMA 806
conversational and OTMA 791
expiration
IMS Connect support 321
OTMA support overview 803
OTMA, specifying in seconds 804
OTMA, specifying in STCK format 804
Fast Path and OTMA 791
flow for standard 826
IMS-to-IMS TCP/IP
communications
transaction code, specifying 780
IMS, using a nonsynchronized tpipe 838
IMS, using a synchronized tpipe 838
OTMA
specify expiration in seconds 804
specify expiration in STCK format 804
terminating conversational transactions 806
transaction expiration overview 803
OTMA grouping 756
protecting 836
recoverable 838
send-then-commit 824, 829
terminating
OTMA conversational transactions 806
unrecoverable 838
transparency option 436
trusted users
IMS Connect 198
tuning
buffers 711
MSC (Multiple Systems Coupling) environment 685
MSC tuning and monitoring 710
two phase commit process 118
two-phase commit
application component 346
application server 346
commit phase 352
communication resource manager (CRM) 346
context token 352
distributed client flow 352, 354
distributed two-phase commit 346
enterprise information system 346
general description 345
global transaction 352
IMS Connect
cross-LPAR support for IMS TM transactions 355
one-phase commit for IMS Universal drivers 349
IMS Connect support 345
IMS Connect support for IMS TM Resource Adapter 352
IMS Connect support for IMS Universal drivers 346
IMS Connector for Java 346
IMS TM Resource Adapter
IMS Connect support 352
one-phase commit optimization 356
IMS Universal Database resource adapter
one-phase commit optimization 349
two-phase commit *(continued)*
IMS Universal Database resource adapters 346
IMS Universal drivers
commit phase 346
context token 346
distributed client flow 346
global transaction 346
IMS Connect support 346
one-phase commit optimization 349
prepare phase 346
prepare phase 352
resource adapter 346
resource manager 346
server distributed syncpoint manager (SDSRM) 346
transaction manager 346
X/Open XA protocol 346
z/OS Resource Recovery Services (RRS) 346
two-phase commit process, definition 28
TYPE macro
defining terminals 384
type-2 connection
WebSphere Application Server Liberty configuration sample 18
type-2 connectivity
WebSphere Application Server Liberty configuration overview 15
type-2 connectivityIMS Universal Database resource adapters
overview 15
type-4 connection
configuration sample
WebSphere Application Server Liberty 14
type-4 connectivity
WebSphere Application Server Liberty configuration overview 12
type-4 connectivityIMS Universal Database resource adapters
overview 12

U

Unaccessed ETO User Control Blocks 98
UNBIND command, stopping session initiation 499
unconditional bracket termination, IMS error handling 948, 949
Unicode
IMS Connect 359
unit of recovery
definition 28
in-flight, definition 28
indoubt, definition 28
UNITYPE keyword on TYPE macro 407
Universal drivers
type-2 connectivity
CICS, configuring 19
UNIX pipes 763
UNIX System Services
MAXFILEPROC parameter 305
socket limits 305
unprotected screen option 436
unrecoverable transactions 838
user
sysplex, in a
recovery status 400

USER 460
user abend 119ABEND 830
user authorization processing 116
user descriptor 86
user ID
  IMSplex, in an
    recovery status 400
User ID caching scheme 807
user message exits
  description and structures 231
  support 213
user workstation
  bracket protocol 515
user workstation.
  API (application program interface)
    CICS asynchronous 623
    CICS synchronous 623
user-data section of OTMA message prefix 886
user-written client application
  IMS Connect message structures 233
users
  operating modes 394
USSTAB option defined in VTAM 419
USTOPPED state 422

V

verification
  IMS terminal support 383
  remote destinations 684
  transaction definitions across systems 703
vertical partitioning 667
Virtual Telecommunications Access Method (VTAM) 367
VLVB records 561
VTAM
  defining the network 415
  MSC (Multiple Systems Coupling) physical link type 667
  MSC and VTAM Generic Resources 730
  MSC physical link
    defining 697
  RNR (Rapid Network Reconnect)
    planning 409
  SLU P
    commands 947
    indicators 947
VTAM (Virtual Telecommunications Access Method)
  ACBNAME parameter 697
  attached terminals 383
  BB indicator 467, 528
  BID command 515, 910
  binary synchronous communications (BSC) 383
  BIS command
    LU 6.1 half sessions 540
    use with CICS 621
  bracket protocol
    CD (change direction) 905
    Finance Communication System 905
    SLU P 905
  bracket protocols 461
  CANCEL command 523, 951
  CD indicator 467, 904
  CHASE command 524
  COMM macro 697
  commands and indicators 456, 902
VTAM (Virtual Telecommunications Access Method) *(continued)*
  devices with MFS support 383
  EB indicator 467, 529
  establishing sessions
    Finance Communication System 904
    SLU P 904
  facilities
    commands and indicators 456, 902
    data transmission 456
    Finance Communication System 902
    IMS 902
    ISC 456
    SLU P 902
    used by ISC 590, 623
  generation
    IMS as host subsystem 416
    LOGON MODE identifiers 418
    NCP buffer pool values 417
    storage requirements 417
    VTAM buffer pool values 417
    VTAM configurations 419
    VTAM nodes 416
  half-duplex protocol, ISC 515
  IMS, relationship to 368
  macros
    SEND 458
    TERMSESS 458
  MSC (Multiple Systems Coupling) linking 667
  network role 367
  node as chosen name 384
  output buffers 417
  parallel sessions 697
  RQR command 904, 951
  SBI command
    LU 6.1 half sessions 540
    use with CICS 621
  SDT command 499
  SESSION parameter 697
  SIGNAL command 540, 951
  synchronous data link control (SDLC) 383
  UNBIND command 499
VTAM Generic Resources
  Multiple Systems Coupling (MSC) 730
VTAM network administration 363
VTAM support 756

W

WAIT_R status
  IMS restart 791
WebSphere Application Server
  configuring
    IMS Universal Database resource adapter, installing 5
  connection factory
    defining 6
    properties 6
  IMS Universal Database resource adapter
    application, installing 7
  IMS Universal drivers
    configuration 4
WebSphere Application Server for distributed platforms
  configuring IMS Universal drivers 3
WebSphere Application Server for z/OS

WebSphere Application Server for z/OS *(continued)*
configuring
type-2 IMS Universal Database resource adapter, installing 8
configuring IMS Universal drivers 3
IMS Universal Database resource adapter
application, installing 11
classpath, setting 9
data source, installing 10
type-2 connectivityIMS Universal Database resource adapters
overview 7
type-2 IMS Universal drivers
configuring 3
WebSphere Application Server Liberty
type-2 connection
configuration file 18
type-4 connection
configuration file 14
work areas, creating for ESAP 131
work unit
backed out 491, 636
CICS 636
committed 491, 636
definition 491
example 492
pending, unilateral decisions 493
status at session initiation 498
workload distribution, MSC 686
workstation, user
API
CICS asynchronous 589, 590
CICS synchronous 589, 590
bracket protocol
input messages, ISC 515
output messages, ISC 518
definition as logical unit 463
logical unit definition 367
terminology 463
workstations
Finance Communication System 900
SLU P 900

X

XCF (z/OS cross-system coupling facility)
basic OTMA message flow 763
XML
conversion to COBOL 212
IMS Connect
converting XML to COBOL 209
message structures 210
XML conversion example 212
IMS Connect XML conversion support
overview 154
XRF
SLU P 919
XRF (Extended Recovery Facility)
APPC/IMS 44, 52
Class 1 terminals 409
Class 2 terminals 407
Class 3 terminals 408
class of service 408
XRF (Extended Recovery Facility) *(continued)*
classes of terminals 406
definition 405
establishing communication
Finance Communication System 905
ISC 489
SLU P 905
system takeover considerations 905
master terminals 393
MNPS 406
processing with OTMA 785
recovery modes 399
SLU P application program 901
takeover considerations
Finance Communication System 905
SLU P 905
terminal support 385, 408
terminals in 406

Z

z/OS
IMS Connect support for Sysplex Distributor 155
Sysplex Distributor, IMS Connect support 155
system log 375
z/OS application programs
accessing IMS databases using ODBA (Open Database Access) 747
accessing IMS databases using Open Database Access (ODBA) 747
z/OS cross-system coupling facility (XCF)
macros and OTMA 791
z/OS program 754
z/OS Resource Recovery Services
and protected transactions 836
exits supported by IMS 836
z/OS Resource Recovery Services (RRS) 27
z/OS Sysplex Distributor
IMS Connect support for 155
Z2 field in message prefix 438

Product Number: 5635-A06
5655-DS5
5655-TM4

SC27-6783-00