IMS
15.2.0

Exit Routines
(2024-08-30 edition)

Note

Before you use this information and the product it supports, read the information in “Notices” on page 783.

2024-08-30 edition.

This edition applies to IMS 15 (program number 5635-A06), IMS Database Value Unit Edition, V15.02.00 (program number 5655-DS5), IMS Transaction Manager Value Unit Edition, V15.02.00 (program number 5655-TM4), and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this information

These topics provide reference information for the exit routines that you can use to customize IMS database, system, transaction management, IMSplex, Base Primitive Environment (BPE), Common Queue Server (CQS), and IMS Connect environments.

This information is available in IBM® Documentation.

Prerequisite knowledge

Before using this book, you should have knowledge of either IMS Database Manager (DB) or IMS Transaction Manager (TM), including the access methods used by IMS. You should also understand basic z/OS® and IMS concepts, your installation's IMS system, and have general knowledge of the tasks involved in project planning.

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

For most IMS library PDF publications, information that is added or changed after the PDF publication is first published is denoted by a character (revision marker) in the left margin. The *Program Directory* and *Licensed Program Specifications* do not include revision markers.

Revision markers follow these general conventions:

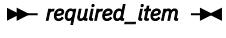
- Only technical changes are marked; style and grammatical changes are not marked.
- If part of an element, such as a paragraph, syntax diagram, list item, task step, or figure is changed, the entire element is marked with revision markers, even though only part of the element might have changed.
- If 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.

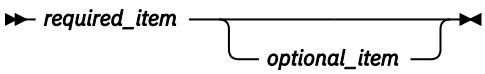
How to read syntax diagrams

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

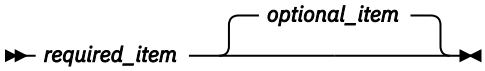
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol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

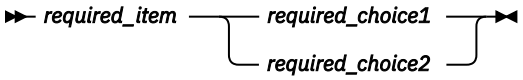


If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

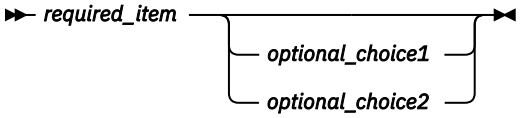


- If you can choose from two or more items, they appear vertically, in a stack.

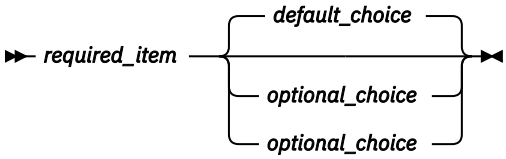
  If you *must* choose one of the items, one item of the stack appears on the main path.



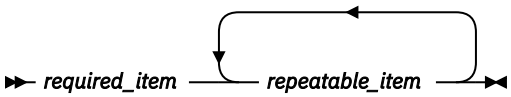
  If choosing one of the items is optional, the entire stack appears below the main path.



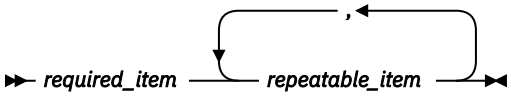
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



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

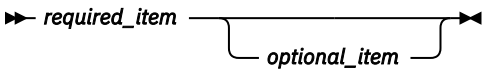


  If the repeat arrow contains a comma, you must separate repeated items with a comma.



  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.

  fragment-name



- 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 15.2

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 15.2. 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 15.2 ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the IMS 15.2 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 15.2 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.
- 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. IMS control region exit routines

Use these topics to design and write user-supplied modules for exit routines that are supported by IMS interfaces and callable services.

Chapter 1. Guidelines for writing IMS exit routines

Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

Introduction to IMS exit routines

Exit routines that customize IMS must adhere to specific guidelines. Use these guidelines when writing an IMS exit routine, when using the callable services that IMS provides for these exit routines, and when analyzing the callable service return and reason codes.

What you can customize

Using IMS-supplied exit routines, you can customize IMS to:

- Edit messages
- Check security
- Edit transaction code input, message switching input, and physical terminal input and output
- Perform additional clean-up
- Initialize dependent regions
- Control the number of buffers the RECON data sets use
- Keep track of segments that have been updated

You can write or include additional routines to customize your IMS system.

Many sample exit routines with default settings are provided in the IMS.SDFSSMPL and IMS.ADFSSMPL libraries.

Related Reading: For information on how to prevent your exit routines from impacting z/OS system integrity, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

You can replace a default exit routine that does not meet your needs by writing one of your own. If you use IMS macros in your exit routine, you must reassemble the routine with the current release level macro library.

Exit routine naming conventions

Each routine name should adhere to naming conventions, including both standard z/OS conventions, and conventions that are specific to the routine.

Using standard z/OS conventions, each routine can have any name up to 8 characters in length. Be sure that this name is unique and that it does not conflict with the existing members of the data set into which you place the routine. Because most IMS-supplied routines begin with the prefix "DFS", "DBF", "DSP", "DXR", "BPE"," CQS", or "CSL", do not choose a name that begins with these letters, unless the specific routine requires it. Also, specify one entry point for the routine.

Naming requirements or exceptions that are specific to an exit routine are noted in the "Naming the Routine" topic of each exit routine section.

Changeable interfaces and control blocks

The interfaces that IMS supplies for use by the exit routines, including the ISWITCH macro, might change in future releases of IMS. IMS control blocks might also change. Therefore, if you write an exit routine that uses these services or control blocks, you might need to change or reassemble the routine accordingly when you migrate to a new release of IMS.

This topic contains Diagnosis, Modification, and Tuning information.

These control blocks include:

DMB
Data management block

PST
Partition specification table

SCD
Systems content directory

VTCB
VTAM® terminal control block

Refreshable exit routine types

For certain types of exit routines, you can designate them as a refreshable exit routine type, which also allows you to call multiple exit routines of that type at the same exit point. These exit routines can be used with the REFRESH USEREXIT command to obtain a new copy of an exit routine without bringing down and restarting IMS.

You can define exit routines for the exit routine types in the EXITDEF parameter in the USER_EXITS section of the DFSDF *xxx* member. The QRY USEREXIT command is used to query information about the routines for the user exit types, and the REFRESH USEREXIT command is used to dynamically refresh the exit routine types. There are no name restrictions for an exit routine that is associated with a refreshable exit routine type.

If an exit is defined in IMS with an EXITDEF statement and as a named module in the STEPLIB concatenation, IMS loads and uses the exit defined in the EXITDEF statement and ignores the module in the STEPLIB concatenation. For example, if the DFSDFxxx member has a USER_EXITS section with EXITDEF=(TYPE=PPUE,EXITS=(DFSPPEX0)) and DFSPPUE0 is in a library in the STEPLIB concatenation, IMS loads and uses DFSPPEX0 as the PPUE user exit. DFSPPUE0 in the STEPLIB concatenation is ignored.

If an exit routine type is not designated as refreshable, you can call only one exit routine of that type and typically the name of the exit routine is designated by IMS.

If an exit routine type is defined as refreshable, multiple exit routines of the same type can be called in sequence. However, any one of the exit routines in the sequence can bypass the remaining subsequent exit routines and return control to the IMS system by setting the SXPLCNXT exit parameter to SXPL_CALLNXTN.

IMS supports the following exit routine types:

- Build Security Environment User Exit (BSEX)
- IMS CQS Event user exit (ICQSEVNT)
- IMS CQS Structure Event user exit (ICQSSTEV)
- IMS Monitor user exit (IMSMON)
- Initialization/Termination user exit (INITTERM)
- Log Edit User Exit (LOGEDIT)
- Logger User Exit (LOGWRT)
- Non-Discardable Messages User Exit (NDMX)
- OTMA Input/Output Edit user exit (OTMAIOED)
- OTMA Destination Resolution user exit (OTMAYPRX)
- OTMA Resume TPIPE Security user exit (OTMARTUX)
- Partner Product user exit (PPUE)
- Program Creation user exit (PGMCREAT)
- Resource Access Security user exit (RASE)
- Restart user exit (RESTART)

- Type-2 Automated Operator User Exit (AOIE)

Related reference

USER_EXITS section of the DFSDFxxx member (System Definition)

REFRESH USEREXIT command (Commands)

“IMS standard user exit parameter list” on page 5

Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

IMS standard user exit parameter list

Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

This interface creates a clearly differentiated programming interface (CDPI) between IMS and the exit routine. Part of the interface consists of a standard user exit parameter list. The list contains information such as a pointer to a version number and a pointer to a function-specific parameter list. All standard user exit parameter lists that have the same version number will contain the same parameters. If a new parameter is added, it is added to the end of the parameter list and the version number is increased by one.

There are currently two active versions of the IMS standard user exit parameter list: version 1 and the current version. The Version 6 standard exit parameter list is the current version. In general, IMS exit routines that do not use the Version 1 standard exit parameter list use the Version 6 standard exit parameter list. Refer to the information for each individual exit routine.

Version 1 standard exit parameter list

The version 1 parameter list contains only pointers to the version number and the function-specific parameter list. The following table shows the content of the Version 1 standard exit parameter list. When the user exit routine is called, IMS passes it the address of this list in register 1.

Table 1. Version 1 standard exit parameter list (mapped by DFSSXPL)

Field	Offset	Length	Description
SXPL	X'00'	N/A	DSECT label for the IMS standard user exit parameter list
SXPLVER	X'00'	X'04'	Address of fullword containing version number of standard exit parameter list
SXPLATOK	X'04'	X'04'	Reserved
SXPLAWRK	X'08'	X'04'	Reserved
SXPLFSPL	X'0C'	X'04'	Address of function-specific parameter list
SXPLINTX	X'10'	X'04'	Reserved
SXPLASCD	X'14'	X'04'	Reserved

The following user exit routines use the Version 1 parameter list:

- “Command Authorization exit routine (DFSCCMD0)” on page 319
- “Fast Path Input Edit/Routing exit routine (DBFHAGU0)” on page 156
- “Greeting Messages exit routine (DFSGMSG0)” on page 178
- “Initialization exit routine (DFSINTX0)” on page 183
- “Logoff exit routine (DFSLGFX0)” on page 196
- “Logon exit routine (DFSLGNX0)” on page 199
- “Destination Creation exit routine (DFSINSX0)” on page 147

- “Signoff exit routine (DFSSGFX0)” on page 278
- “Signon exit routine (DFSSGNX0)” on page 281

Version 6 standard exit parameter list

This version is the current version of the parameter list. The Version 6 standard exit parameter list contains additional fields beyond those in version 1 of the parameter list. The following table shows the layout of the parameter list. When a user exit routine is called, IMS passes the address of this parameter list to the exit routine module in register 1.

Table 2. Version 6 standard exit parameter list (mapped by DFSSXPL)

Field	Offset	Length	Description
SXPL	X'00'	N/A	DSECT label for the IMS standard user exit parameter list
SXPLVER	X'00'	X'04'	Address of fullword containing version number of standard exit parameter list
SXPLATOK	X'04'	X'04'	0 or the address of a fullword containing the callable services token for this instance of the routine
SXPLAWRK	X'08'	X'04'	Pointer to a 512-byte work area. This area is intended as working storage for a user exit routine. The storage is not initialized, and may contain residual data. The contents of the storage are not guaranteed to be preserved between user exit calls. If a work area that is preserved between calls is required, use the storage pointed to by SXPLASWA.
SXPLFSPL	X'0C'	X'04'	Address of the function-specific parameter list
SXPLINTX	X'10'	X'04'	Address of the user data table loaded by DFSINTX0 at IMS initialization time. This field is valid only in IMS environments where DFSINTX0 is called. It will be X'80000000' in any other environment.
SXPLASCD	X'14'	X'04'	Address of the IMS SCD

Table 2. Version 6 standard exit parameter list (mapped by DFSSXPL) (continued)

Field	Offset	Length	Description
SXPLASWA	X'18'	X'04'	Address of a 256–byte static work area. Each exit routine is assigned its own static work area and is available for the exit routine to store data that is preserved from call to call. The static work area is cleared before the first time the exit routine is called. Each exit routine is assigned a separate static work area that is preserved between calls to that exit routine. This work area is available for all user exits that use this version of the standard exit parameter list, regardless of whether the exit is defined with the EXITDEF parameter in the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. If your exit routine can be called concurrently under different ITASKs, you must consider the results of sharing a single static work area in the design of your exit routine. If an exit routine is replaced with the REFRESH USEREXIT command, the same static work area is passed to the new version of the exit routine. If an exit routine is deleted with the command, the static work area is also deleted. If a new exit routine is added with the command, a new static work area is allocated. The same static work area is shared by the old and new versions of an exit routine until the old exit is deleted. This must be handled by your exit routine in the same way as the multiple concurrent executions under different ITASKs. SXPL_F1SWASHR is set when your exit is called while the static work area is shared. When the old exit routine is deleted, SXPL_F1WASHR is no longer set.
SXPLIMSR	X'1C'	X'04'	Address of the version of IMS that is calling the exit. The 4-byte version data is stored in the following format: `0000`*`vvmm`* ***vv*** IMS version (SSCDIMSR) ***mm*** IMS mod (SSCDIMSM)
SXPLIMID	X'20'	X'04'	Address of the IMS ID. Standalone invocation of the Image Copy Utility and Change Accumulation Utility set address of the job name.
SXPLRSEN	X'24'	X'04'	Address of the 8 character Recoverability Service Name (RSENAME). This name is set using the RSENAME startup parameter in the DFSHSBxx member. If the control region is not XRF capable or DBCTL warm standby capable, this field is blank.

Table 2. Version 6 standard exit parameter list (mapped by DFSSXPL) (continued)

Field	Offset	Length	Description
SXPLCNXT	X’28’	X’04’	Address of a flag byte in storage. The flag indicates if the next exit routine in the definition list will be called after this exit routine releases control. When an exit type is defined as refreshable, multiple exit routines of the same type can be called in sequence. By setting this flag to SXPL_CALLNXTN, an exit routine in the sequence can return control to the IMS system without calling any subsequently defined exit routine. **SXPL_CALLNXTN** The next exit routine will not be called. **SXPL_CALLNXTY** The next exit routine will be called.
SXPLFLGA	X’2C’	X’04’	Address of a full word in storage that contains flags for the user exit. **SXPL_F1ENHSRV** The exit is called with the enhanced callable services, including the ability to call multiple exit routines of the same user exit type. **SXPL_F1SWARFR** Do not use this flag. Instead, use either SXPL_F1RFRSHD, SXPL_F1SWASHR, or both. **SXPL_F1RFRSHD** If this flag is set, the exit is refreshed. This flag is set based on a flag in a control block that represents the exit routine, and this flag is not reset until the exit returns to IMS. If the user exit runs in parallel, multiple calls to the user exit can be made with this flag set before the control block flag is reset. **SXPL_F1SWASHR** If this flag is set, the static work area whose address is contained in SXPLASWA is shared and can be accessed by the old and new versions of the exit because the old version might still be active. You should consider this behavior when you use your user exit if the two versions use different layouts for the area. When the old exit routine is deleted, this flag will no longer be set.

If an exit routine is written to use a parameter that was added in a later version, and the exit routine can execute in an environment in which earlier versions of the parameter list could be received, the exit routine should check the version of the parameter list it receives to ensure that the data is available to the exit routine.

Related reference

“Initialization exit routine (DFSINTX0)” on page 183

Use the Initialization exit routine (DFSINTX0) to create two user data areas that can be used by some of your installation's exit routines.

Using the ISWITCH macro

The ISWITCH macro changes execution from the dependent region TCB to the control or DL/I address space. ISWITCH also exits cross-memory mode. If you executing an ISWITCH macro call, follow the guidelines in this information.

ISWITCH must have addressability to the SCD and, for the following figure, to the PST. The address of the SCD is obtained from the PSTSCDAD field in the PST.

For Fast Path exit routines, specify TO=CTL.

The following figure is an ISWITCH example:

ISWITCH example

```
ISWITCH TO=DLI,ECB=PSTDECB
SLR    R1,R1            Get a zero
ST     R1,PSTDECB       Clean ECB after target memory post
LTR    R15,R15          Successful?
BNZ    ERR1             No
```

When a Fast Path exit routine issues an ISWITCH to the control region, it must issue a second ISWITCH call specifying TO=DEP to return to the dependent region before returning to the caller of the exit routine. This is done only in an exit routine that is entered from a Fast Path module.

The following is an example of the second ISWITCH call needed for Fast Path:

```
ISWITCH TO=DEP,ECB=PSTDECB
SLR    R1,R1            Get a zero
ST     R1,PSTDECB       Clean ECB after target memory post
LTR    R15,R15          Successful?
BNZ    ERR1             No
```

Exit routines should not use ISWITCH TO=RET, because unpredictable results might occur. (ISWITCH TO=RET could be used in previous IMS releases.) Ensure that all instances of ISWITCH TO=RET are changed to ISWITCH TO=DEP.

Routine binding restrictions

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Most modules receive control and must return control in AMODE=31, and must be able to execute in cross-memory and TASK modes.

Recommendations:

- RMODE=ANY is recommended.
- All TM exit routines can be entered simultaneously by multiple dispatchable tasks. Therefore, it is highly recommended that all TM exit routines are coded as *reentrant* (RENT).

All routines receive control and must return control in 31-bit addressing mode (AMODE 31) and must be able to execute in RMODE ANY and AMODE 31.

If you bind an exit routine as reentrant (RENT), it must be truly reentrant (for example, it cannot depend on any information from a previous iteration and it cannot store into itself).

If you bind an exit routine as reusable (REUSE), it must be truly reusable (it cannot depend on any information in itself from a previous iteration), but it can depend on information that it saves in the specific block passed to it. If you bind a routine that is serially reusable, it must be used for a single database only.

If you bind an exit routine as neither RENT nor REUSE, it can store into itself and depend on the information saved in the block that is passed to it.

If you bind an exit routine as reentrant, it is loaded in key 0 storage to automatically protect the exit routine from being accidentally or intentionally modified.

Unless otherwise indicated for a particular exit, all IMS user exits called from the IMS Control, DLI, DBRC, dependent (including DBCTL threads), and IMS batch regions should be bound in a PDS data set included in the JOBLIB or STEPLIB DD concatenation. Exits in PDSEs are not supported.

Specific requirements and exceptions are noted in each topic. Refer to the topic on "Binding the Routine" included in each exit routine section.

Writing IMS routines that access control blocks

Control blocks for databases, programs, transactions, and routing codes are not in contiguous storage. This is true whether dynamic resource definition is enabled or not. If you have exit routines that depend on these resources being in contiguous storage, you will have to change them.

These requirements apply specifically to:

- DMB directory entries (DDIR)
- PSB directory entries (PDIR)
- Routing code table entries (RCTE)
- Scheduler message blocks (SMB)

If your routine accesses IMS control blocks, you can find DSECTs for these blocks in the following macros:

Macro
DSECT

ISCD
System content directory (SCD)

DFSDDIR
DMB Directory entry (DDIR)

DFSPDIR
PSB Directory entry (PDIR)

DFSDMB
Data management block (DMB)

DFSPSB
Program specification block (PSB)

DBFESCD
Extended system content directory (ESCD)

DBFRCTE
Routing code table entry (RCTE)

IAPS
Scheduler message block (SMB)

Extended Terminal Option (ETO) exit routines

Unless otherwise stated, all non-LU 6.2 exit routines are available to terminals that are defined both statically at system definition and dynamically by using the Extended Terminal Option (ETO) feature.

Some exit routines are loaded at initialization if ETO=Y. (If this is the case, it is noted in each in the topic on Binding or including the routine.) Although these exit routines are loaded only if the ETO feature is used, they are available for use by static and dynamic ACF/VTAM terminals.

Related Reading: For more information about ETO, see *IMS Version 15.2 Communications and Connections*.

APPC/IMS exit routines

Some exit routines support LU 6.2 devices and are affected by APPC/IMS.

The LU 6.2 Edit exit routine (DFSLUEE0) is available only to LU 6.2 devices. The following exit routines also support LU 6.2 devices:

- Message Control Error exit routine (DFSCMUX0)
- Conversational Abnormal Termination exit routine (DFSCONE0)
- Transaction Authorization exit routine (DFSCTRN0)
- Fast Path Routine for Input Edit/Routing exit routine (DBFHAGU0)
- Command Authorization exit routine (DFSCCMD0)
- TM and MSC Message Routing and Control User exit routine (DFSMSCE0)

No other exit routines support LU 6.2 devices or are affected by APPC/IMS.

Registers and save areas

IMS exit routines need to save registers in the save area pointed to by Register 13. This save area is provided at entry. In general, the save area passed to the exit is in 31-bit storage. You should save and restore registers in 31-bit mode.

There are two types of save areas that exit routines use to save registers:

- A prechained save area passed to the exit routine by IMS or the calling application
- A single save area used by exit routines that use the Version 5 standard user exit parameter list

Using the prechained save area

IMS or the application that calls the exit routine passes a prechained save area to the exit. The routine must step forward to the next save area in the save area set before processing any data.

The save area address given to the exit routine has a prechained forward save area pointer at offset 8 and a prechained backward pointer at offset 4. The exit routine can use the forward save area pointed to by offset 8 but must not alter the first three words of the save area.

Before returning control to IMS, the routine must step back to the original save area and restore IMS registers.

Using the single save area

When an exit routine uses the version 5 standard user exit parameter list, it does not receive a prechained save area. Instead, the routine points to a single save area in register 13. The exit routine must use this save area to save registers from IMS or the calling application.

If the exit routine calls other applications or routines, including IMS callable services, the routine must provide an additional save area. The 512-byte dynamic work area passed to exit routines that use the Version 6 standard exit parameter list can be used as one or more save areas.

Before returning control to IMS, the exit routine must restore the registers to IMS or the calling application.

Related reference

“IMS standard user exit parameter list” on page 5

Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Cross-memory considerations

Restrictions exist which should be considered when writing an IMS exit routine that will perform while in cross-memory mode.

Do not issue any SVC (except ABEND) or I/O request.

If the routine runs in the DL/I address space and you need to perform a function that cannot be done in cross-memory mode, issue an ISWITCH TO=DLI to exit cross-memory. Because of the overhead in performing a task switch from the dependent address space to the IMS control program, use ISWITCH infrequently. ISWITCH TO=DLI is not valid for TM exit routines.

If you are not using the DL/I address space option, execution after the ISWITCH continues in the control address space. With LSO=S, execution continues in the DL/I address space. TO=DLI on ISWITCH performs the correct switching in all environments.

With LSO=S, DL/I exits cannot address data in the control address space.

Most terminal-related control blocks are not addressable from the DL/I address space.

Sometimes your exit might need to test to determine whether it is running in cross-memory mode before making a particular function call. In IMS, when an exit is called in cross-memory mode, the primary address space will always be different from the secondary address space. You can use the instructions EPAR and ESAR to obtain the primary and secondary address space ASIDs and compare them. If they are equal, the exit is not in cross-memory mode. If they are unequal, the exit is in cross-memory mode.

The following sample shows an example of checking for running in cross-memory mode. The code issues a branch-enter WTO macro call when it is in cross-memory mode; it issues a normal SVC WTO when not in cross-memory mode.

```
         EPAR  R0             Get primariy ASID
         ESAR  R1             Get secondary ASID
         CLR   R0,R1          Primary = Secondary?
         BNE   BEWTO          No, in XM mode, use BE WTO
         WTO   'message'      Yes, use SVC WTO
         B     ENDWTO
BEWTO    DS    0H
         WTO   'message',LINKAGE=BRANCH
ENDWTO   DS    0H
```

Exit routine performance recommendations

Efficiency of exit routines is a prime concern for IMS performance. The amount and type of processing that is done by exit routines can directly contribute to the total path length and time required to complete a unit of work.

Most routines are called from the IMS control region and get control in key 7 supervisor state. Some routines might be called from mainline processing code running under the IMS Control Region task. Other units of work that must wait to run under a task currently in use by an exit routine can also be affected. An abend in an exit routine that gets control in the IMS control region can cause the IMS control region to abend.

Recommendations:

- Code user-written routines in ways that minimize path length and processing time as much as possible.
- Use services such as OS WAITs, SVCs, and I/O sparingly. When an IMS callable service exists, use it rather than the z/OS equivalent. The IMS callable service is optimized to perform more efficiently in an IMS subdispatching environment.
- Write IMS exit routines in assembly language rather than high-level languages. IMS does not support exit routines running under Language Environment® for z/OS.

IMS callable services

IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

Types of callable services

IMS callable services may consist of services for storage, control blocks, and the automated operator interface (AOI).

Storage services support the following functions:

- Get storage
- Free storage
- Load module
- Delete module

Control block services support the following functions:

- Find control block
- Scan control block

AOI services support the following functions:

- Insert message
- Enqueue message
- Cancel message

Exit routines eligible for callable services

An exit routine may use one or more of the three types of callable services: storage, control block, and AOI. DFSAOE00 is the only exit routine that is eligible to use AOI callable services.

The following table shows the exit routines that are eligible for callable services and the types of callable service that they can use. See the topic for each exit routine for more information on how it uses callable services.

Table 3. Exit routines and associated callable services

Exit name or user exit type	Callable services		
	Storage	**Control block**	**AOI**
BSEX	X		
DBFHAGU0	X	X	
AOIE	X		X
DFSAOUE0	X	X	
DFSCCMD0	X	X	
DFSCMLR1	X	X	
DFSCMPX0	X	X	
DFSCNTE0	X	X	
DFSCONE0	X	X	
DFSCSGN0	X	X	
DFSCSMB0	X	X	

Table 3. Exit routines and associated callable services (continued)

Exit name or user exit type	Callable services		
	Storage	**Control block**	**AOI**
DFSCTRN0	X	X	
DFSCTSE0	X	X	
DFSCTTO0	X	X	
DFSFEBJ0	X	X	
DFSGMSG0	X		
DFSGPIX0	X	X	
DFSINSX0	X	X	
DFSINTX0	X	X	
DFSI7770	X	X	
DFSLGFX0	X	X	
DFSLGNX0	X	X	
DFSME000	X	X	
DFSME127	X	X	
DFSMSCE0	X	X	
DFSO7770	X	X	
DFSPIXT0	X	X	
DFSQSPC0	X	X	
DFSSGFX0	X	X	
DFSSGNX0	X	X	
DFSSIML0	X	X	
DFSS7770	X	X	
DFSYPRX0	X	X	
DFSYIOE0	X	X	
DFSYDRU0	X	X	
DFSYRTUX	X	X	
DFS29800	X	X	
DFS36010	X	X	
LOGWRT	X		
NDMX	X		
PGMCREAT	X		
PPUE	X		

Using callable services

You will need to initialize callable services for your IMS exit routine each time that your exit routine gets control.

To use a callable service, do the following:

1. Link your exit routine to the callable service interface module (DFSCSI00).
2. Initialize callable services for your exit routine (CALL DFSCSII0) each time your exit routine gets control.
3. Initialize the callable services parameter list.
4. Initialize the function-specific parameter list.
5. Invoke the callable service (CALL DFSCSIF0).

Repeat steps 3 through 5 as many times as necessary while your exit routine has control.

Not all exit routines perform all five of the preceding steps. See the section called "Using IMS callable services " in the description of the specific exit routine you are coding to see which steps apply.

Callable services

To use IMS callable services, an exit routine must invoke one of two IMS callable services entry points in AMODE 31. The exit routine will receive a control block and a callable services parameter list.

The callable services interface module DFSCSI00 contains two entry points that your exit routine can invoke: DFSCSII0 and DFSCSIF0.

Entry point DFSCSII0 initializes callable services. To begin initialization, issue CALL DFSCSII0 with the appropriate registers initialized. DFSCSII0 returns a callable services token and a parameter list address. The callable services token must be passed to IMS when you invoke one of the callable services. The parameter list address directs reentrant programs to a storage area in which to build parameter lists needed to invoke callable services.

Entry point DFSCSIF0 invokes one of the callable services. To invoke a callable service, issue CALL DFSCSIF0 with the appropriate information specified. You must tell IMS which service to invoke. You do this by initializing two parameter lists. The first list, the callable services parameter list, contains information needed by callable services to route the request to the appropriate service. The second list, the function-specific parameter list, defines which service is to be used and provides information required by that service.

When your exit routine receives control back from callable services, register 15 contains a return code indicating whether the call was successful. The callable services parameter list contains a return code and a reason code if the call did not complete successfully. The function-specific parameter list can contain data from a specific callable service.

Exit routine assembler macros

You can use assembler macros to generate parameter list DSECTs for your exit routine.

To generate parameter list DSECTs, you can use the following assembler macros in your exit routine.

Macro
Description

DFSCSIPL
Generates the DFSCSPL, DFSCSTRG, DFSCCBLK, and DFSAOI parameter list DSECTs for an exit routine.

DFSCSPL
Generates the callable services parameter list DSECT (CSPARMS).

DFSCSTRG
Generates the storage services function-specific parameter list DSECT (CSSTRG).

DFSCCBLK
Generates the control block services function-specific parameter list DSECT (CSBLK).

DFSAOI
Generates the AOI services function-specific parameter list DSECT (DFSAOI).

DFSLOGP
Generates the function-specific parameter list for the IMS Monitor (IMSMON) exit in DSECT MONEXPL.

Links with your exit routine and DFSCSI00

To use callable services, your exit routine must be linked with the callable service interface module, DFSCSI00. You need to manually link this module to your exit routine.

Manual linking

To use callable services, you must manually link these exit routines to DFSCSI00.

Exit routines or user exit types to be manually linked to DFSCSI00	Exit routines or user exit types to be manually linked to DFSCSI00
DFSAOE00	DFSINTX0
AOIE	DFSLGFX0
BSEX	DFSLGNX0
DFSCCMD0	DFSMSCE0
DFSCSMB0	NDMX
DFSCTSE0	PPUE
DFSGMSG0	DFSSGFX0
DFSGPIX0	DFSSGNX0
DFSINSX0	LOGWRT
DFSCNTE0	DFSFEBJ0
DFSME000	DFSME127
DFSSIML0	

Typically, you must manually link DFSCSI00 if your exit routine is a stand-alone module (not linked as part of another IMS load module). When you perform this binding, include an ENTRY bind control statement that specifies the entry point of your exit routine. The statement ensures that your exit routine, and not DFSCSI00, receives control when IMS calls it.

Initialization of IMS callable services (DFSCSII0)

Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Exit routines that do not receive the IMS standard user exit parameter list (DFSSXLP) in register 1 on entry, or that do receive DFSSXPL but with a zero value for field SXPLATOK, must initialize IMS callable services.

Exit routines that receive DFSSXLP in register 1 with a non-zero value for field SXPLATOK do not need to initialize callable services. These routines should use the callable services token referenced in SXPLATOK for all calls to IMS callable services. A routine that receives a token can use the work area pointed to in SXPLAWRK to get the callable services parameter list.

The callable services token is used to request a specific callable service through a subsequent call to entry point DFSCSIF0.

The parameter list that is returned in register 1, contains the callable services token. You need to extract the token and save it, so it does not get overlaid. Then the parameter list can be formatted for your callable service request. The parameter list is large enough to contain the parameter lists that accompany your request.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with your exit routine. The contents of register 0 are not preserved on entry and exit.

The following two tables list the content of registers on entry and return to and from DFSCSII0.

Content of registers on entry to DFSCSII0

Register	Content
1	ECB Address. On entry, IMS gives the address of an ECB to each exit routine that can issue callable service requests. The ECB address must be passed on the DFSCSII0 initialization call. See the section for each exit routine to determine where to find the ECB address for that exit routine.
13	Address of save area for use by DFSCSII0.
14	Caller's return address.
15	DFSCSII0 entry point address.

Content of registers on return from DFSCSII0

Register	Content
1	Address of parameter list **Offset** **Description** **0** Callable services token, which is four bytes long.
15	Return code **Return code** **Meaning** **0** Request was successful. **4** Callable services are unavailable. **8** Callable services are unavailable. Initialization failed due to insufficient storage. **12** Callable services are unavailable. Initialization failed due to errors in IMS control blocks.

Callable services parameter list

CSPARMS is the callable services parameter list required for all callable service requests. Callable services use parameters in the list to route control from the module requesting the service to the service

routine that processes the request. The list is also used to pass return and reason codes from the service to the exit routine.

Initialize the parameter list with the callable services token and the code of the callable service you want to use (storage services, control block services, or AOI services). All other fields should be cleared. If the exit routine issues multiple calls, you can save the callable services token in a register and restore it to CSPLTOKN on subsequent calls.

Initialize the following fields:

Field	Offset	Length	Description
CSPLTOKN	0	4	IMS callable services token
CSPLSERV	4	4	IMS callable service code. The values are as follows: **1** Storage services **2** Control block services **3** AOI services

Function-specific parameter list initialization

After specifying which service you want to use in the callable services parameter list, indicate which function of the service you want to use by initializing the appropriate function-specific parameter list.

Related reference

"IMS Callable Storage Services" on page 20
CSSTRG is the function-specific parameter list used for IMS Callable Storage Service requests. It is defined by the DFSCSTRG macro.

"IMS Callable Control Block Services requests" on page 23
CSCBLK is the function-specific parameter list used for IMS Callable Control Block Service requests. It is defined by the DFSCCBLK macro.

"IMS Callable AOI Services" on page 28
DFSCAOI is the function-specific parameter list used for IMS Callable AOI Service requests. The DFSCAOI macro defines these requests.

IMS callable service (DFSCSIF0) activation

IMS uses the entry registers, parameter lists, and exit registers to communicate with your exit routine.

Communicating with IMS

To activate a callable service, issue `CALL DFSCSIF0 (callable services parameter list, function-specific parameter list)`.

The following tables list the content of registers on entry and exit to and from DFSCSIF0.

Table 4. Content of registers on entry to DFSCSIF0	
Register	**Content**
1	Address of two-word parameter list built by CALL macro. **Offset** **Description** **0** Callable services parameter list address **4** Function-specific parameter list address
13	Address of save area for use by DFSCSIF0
15	DFSCSIF0 entry point address

Table 5. Content of registers on return to DFSCSIF0	
Register	**Content**
15	**Return code** **Meaning** **0** Request successful **4** Request unsuccessful

If the request is unsuccessful, refer to the return (CSPLRTRN) and reason code (CSPLRESN) fields in the callable services parameter list described in the following table.

Table 6. Content of registers on return from DFSCSIF0	
Field	**Description**
CSPLRTRN	Return code set with error codes defined in DFSCSPL. For a list of these codes, refer to "Return codes (CSPLRTRN)" on page 30.
CSPLRESN	Reason code set with error codes defined in DFSCSPL. For a complete description of the reason codes, see one of the following sections: **Reason code** **Reference** **4** See "Callable service interface reason codes (CSPLRESN)" on page 30. **8** See "Function-specific parameter list reason codes (CSPLRESN)" on page 31.

IMS Callable Storage Services

CSSTRG is the function-specific parameter list used for IMS Callable Storage Service requests. It is defined by the DFSCSTRG macro.

The function-specific parameter list contains the information that storage services need to perform the function you requested (get or free storage, load or delete a module). The function-specific parameter list is also used to return data to the exit routine.

You must initialize the function-specific parameter list for storage services before calling DFSCSIF0 to activate storage services. All fields that are not used as input to DFSCSIF0 should be cleared.

GET storage function

You can obtain user storage for any IMS exit routine that uses IMS callable services by initializing the GET storage function in CSSTRG.

The storage can be obtained in private storage or CSA with either doubleword or page boundary alignment. The storage can be requested above (31-bit) or below (24-bit) the 16 MB line.

To request the GET storage function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: 1 = GET storage
CSGTLEN	4	4	Length of storage to obtain
CSGTSP	8	4	Storage subpool identifier values: • 0 = private storage • X'FFF' = CSA storage
CSGTLOC	C	4	Storage location identifier values: • 0 = 31-bit storage • 1 = 24-bit storage
CSGTBNDY	10	4	Storage boundary identifier values: • 0 = doubleword boundary • 1 = page boundary

The following field (in CSSTRG) is returned from the GET storage function:

Field	Offset	Length	Description
CSGTADDR	14	4	Storage address

FREE storage function

You can release user storage previously obtained by the GET storage service by using the FREE storage function.

The requestor specifies the address of the storage service. The storage subpool (private or CSA) specified on the FREE request must be the same value specified on the GET request.

To request the FREE storage function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: • 2 = FREE storage
CSFRSTAD	4	4	Address of storage to release
CSFRLEN	8	4	Length of storage to release
CSFRSP No data is returned from the FREE storage service.	C	4	Storage subpool identifier values: • 0 = private storage • X'FFF' = CSA storage

LOAD module function

You can load a module for any IMS exit routine that uses IMS callable services by initializing the LOAD module function in CSSTRG.

The module can be loaded in private storage or CSA. The module can be loaded above (31-bit) or below (24-bit) the 16 MB line. The name of the module must be specified. If the module was loaded previously but you want a new copy of the module, you can request a load of a new copy.

The LOAD module function can be requested by callers running in cross memory mode. In this case, the LOAD module function determines if the primary address space is either CTL or DLI/SAS, and ensures that the call executes in the proper address space in non-cross memory mode. The LOAD module function restores the cross memory environment before returning control to the caller.

There might be a noticeable performance impact for cross memory callers issuing the LOAD module function, because this call requires that the environment be switched from cross memory mode to non-cross memory mode and then restored. Use of the LOAD module function should be kept to a minimum for mainline path exit routines.

To use the LOAD module function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: • 5 = LOAD module
CSLDNAME	4	8	Name of module to load
CSLDSP	C	4	Storage subpool identifier values: • 0 = private storage • X'FFF' = CSA storage
CSLDLOC	10	4	Module storage location identifier values: • 0 = 31-bit storage • 1 = 24-bit storage
CSLDUSE	14	4	Module reuse identifier values: • 0 = use existing copy of module if found • 1 = load a new copy of module

The following fields are returned from the LOAD module function:

Field	Offset	Length	Description
CSLDMEP	18	4	Module entry point
CSLDMLEN	1C	4	Module length bit 0 is set to one when the module was previously loaded

DELETE module function

You can use the DELETE module storage service to delete a module previously obtained by the LOAD storage service.

The requester specifies either the module name or module address. If more than one copy of the module was loaded, the address should be used instead of the name to ensure that the correct copy is deleted. The module storage subpool (private or CSA) specified on the DELETE request must be the same value specified on the LOAD request.

The DELETE module function can be requested by callers running in cross memory mode. In this case, the DELETE module function determines if the primary address space is either CTL or DLI/SAS, and ensures that the call executes in the proper address space in non-cross memory mode. The DELETE module function restores the cross memory environment before returning control to the caller.

There might be a noticeable performance impact for cross memory callers issuing the DELETE module function, because this call requires that the environment be switched from cross memory mode to non-cross memory mode and then restored. Use of the DELETE module function should be kept to a minimum for mainline path exit routines.

To request the DELETE module function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: 6 = DELETE module
CSDLNAME	4	8	Name of module to delete. Either module name or module address must be specified to delete a module. The unused field should be cleared. If the module name is not specified, this field should be cleared and CSDLEP must be specified.
CSDLEP	C	4	Address of module to delete. If more than one copy of the module was loaded, delete the module by specifying the module entry point. This ensures that the correct copy of the module is deleted. If both name and address are specified, the module is deleted using the address. If the address is not given, the name must be specified and all copies will be deleted.
CSDLSP	10	4	Storage subpool identifier values: 0 = private storage X'FFF' = CSA storage

No data is returned from the DELETE module function.

IMS Callable Control Block Services requests

CSCBLK is the function-specific parameter list used for IMS Callable Control Block Service requests. It is defined by the DFSCCBLK macro.

The function-specific parameter list contains the information control block services need to perform the function you requested (find or scan a control block). The function-specific parameter list is also used to return data to the exit routine.

Restriction: Global terminal or user resource information is not available to user exit DFSLGNX0. Callable services will only return local information for DFSLGNX0.

If an IMSplex is sharing terminal or user information in Resource Manager (RM), callable services automatically and transparently return global resource information to the exit routine. However, if a routine scans resources that are only local to the current IMS, it can specify the local option (by setting CSFDFLG1). For resources that do not have global information such as transactions, the local option results in the same information as the default.

You must initialize the function-specific parameter list for control block services before calling DFSCSIF0 to activate control block services. All fields that are not used as input to DFSCSIF0 should be cleared.

FIND control block function

You can find a specific instance of a control block within any IMS exit routine that uses IMS callable services by initializing the FIND control block function in the DFSCCBLK macro.

The search type identifies the type of control block to locate. A search type can include more than one type of control block. A list of the search types is in the description of the CSFDTYPE field in the following table. The control block name or identifier is used to find a specific instance of the control block.

Initialize the function-specific parameter list before calling DFSCSIF0 to activate control block services. All fields that are not used as input to DFSCSIF0 should be cleared.

Initializing the function-specific parameter list for FIND

In all instances, you need to initialize the following three fields:

Field	Offset	Length	Description
CSCBFUNC	0	1	IMS control block services function code value: • 1 = FIND control block
CSFDTYPE	4	4	Control block search type values: 1 = FIND CCB 2 = FIND CNT, or LNB 3 = FIND RCNT 4 = FIND CNT, LNB, or RCNT 5 = FIND SPQB 6 = FIND VTCB 7 = FIND CNT descriptor 8 = FIND LOGON descriptor 9 = FIND USER descriptor 10 = FIND transaction 11 = FIND LLB for MSC MSLINK 12 = FIND LCB for MSC MSPLINK

Field	Offset	Length	Description
CSFDFLG1	16	1	Input Flag Byte **CSFDLOC1 EQU X'80'** Indicates that the FIND request is to return local information only.

Depending on the type of block you want to find, you must initialize the following fields:

Block type to find	Field to initialize

CCB	Specify one of the first two fields, and clear the unused field. The LTERM name field must always be specified.
	`CSFDEID = EBCDIC CCB identifier`
	`CSFDBID = Binary CCB identifier`
	`CSFDNAME = associated LTERM name`

CNT or LNB	Use the LTERM name to locate a specific CNT or LNB. `CSFDNAME = LTERM name`
RCNT	Use the LTERM name to locate a specific RCNT. `CSFDNAME = LTERM name`
CNT, LNB, or RCNT	Use the LTERM name to locate a specific CNT, LNB, or RCNT. `CSFDNAME = LTERM name`
SPQB	Use the USER name to locate a specific SPQB. `CSFDNAME = USER name`

VTCB	Either the node name alone or the node and user name are used to locate a specific VTCB. If the user name is not used on the request, clear the unused field.
	`CSFDNODE = Node name`
	`CSFDUSER = User name`

CNT, LOGON, or USER Descriptor	Specify the name of the descriptor you want to locate. `CSFDNAME = CNT, LOGON, or USER descriptor name`
Transaction	Specify the transaction code you want to find. `CSFDNAME = Transaction name`

LLB	Use the Link number or Link name of the MSLINK to locate a specific instance of an LLB.
	`CSFDNAME = Logical Link name or Link number`
	If Link name, use an 8-character name.
	If Link number, format must be 2 words. The first word is the binary link number. The second word is zero. So Link 1 format would be:
	`CSFDNAME = 00000001 00000000`

Block type to find	Field to initialize
LCB	`CSFDNAME = Use the 8-character Physical link name`

Output returned from FIND Control Block Services

Depending on the type of search specified, one of the following is returned in the CSFDBLKA field in the function-specific parameter list:

Search type	Output from service
FIND CCB	CCB address
FIND CNT or LNB	CNT or LNB address
FIND RCNT	RCNT address
FIND CNT, LNB, or RCNT	CNT, LNB, or RCNT address
FIND SPQB	SPQB address
FIND VTCB	CLB address
FIND CNT descriptor	USRD address
FIND LOGON descriptor	CLB address
FIND USER descriptor	USRD address
FIND Transaction	SMB address FIND transaction also returns the PDIR address in field CSFCBLK2.
FIND LLB	LLB address
FIND LCB	LCB address

SCAN control block function

You can use the SCAN control block function to scan control blocks of a certain type.

The first time the SCAN function is activated, the current control block address should be 0. SCAN returns the first control block that meets the search criteria. The SCAN function can be subsequently activated to locate additional control blocks. Subsequent searches start where the previous scan left off.

On subsequent SCAN requests, the current block address is passed back to the service. The search starts with the current control block to locate the next control block meeting the criteria. The blocks are not retrieved in alphabetic sequence.

Subsections:

- "Qualifying the scan" on page 25
- "Initializing the function-specific parameter list for SCAN" on page 26
- "Output returned from SCAN Control Block Services" on page 27

Qualifying the scan

To further qualify the scan, a generic name or a name containing wild cards can be specified for CNT, LNB, RCNT, SPQB, and VTCB control block types.

- A generic name consists of one or more characters of the name followed by an asterisk. Generic names must be padded with blanks.

For example, assume valid names are DFSAAAAA, DFSZZZZZ, and DFSABBBB. Multiple scan requests using the generic name 'DFSA*' can be used to obtain the control block addresses for DFSAAAAA and DFSABBBB. In this case, DFSZZZZZ would not be returned to the requester.

- A wild card character is represented by the '%' character. One or more wild cards can replace characters within the name when that position in the name can be any character.

  For example, assume valid names are DFSAABBB, DFSZZBBB, and DFSABCDE. Multiple scan requests using the name DFS%%BBB containing wild card characters in positions 4 and 5 would return control block addresses for DFSAABBB and DFSZZBBB. DFSABCDE would not be returned to the requester.

You must initialize the function-specific parameter list before calling DFSCSIF0 to activate control block services. All fields that are not used as input to DFSCSIF0 should be cleared.

Initializing the function-specific parameter list for SCAN

To request a SCAN and search type, you always need to initialize the first two fields as follows:

Field	Offset	Length	Description
CSCBFUNC	0	4	IMS control block service function code value: • 2 = SCAN control block
CSSCTYPE	4	4	Control block search type indicator values: 1 = SCAN CCB 2 = SCAN CNT or LNB 3 = SCAN RCNT 4 = SCAN CNT, LNB, or RCNT 5 = SCAN SPQB 6 = SCAN VTCB 7 = not used 8 = SCAN LOGON descriptor 9 = SCAN USER descriptor 10 = not used SCAN LLB SCAN LCB
CSSCFLG1	20	1	**CSSCLOC1 EQU X'80'** Indicates that the SCAN request is to return local information only

Depending on the type of search you want, you might also need to initialize one or more of the following fields in the function-specific parameter list.

To scan	Initialize
CCB	Specify whether you want to scan for the first CCB or to start the scan at the current CCB. `CSSCCBLK = Current CCB address or zero`

CNT or LNB	Specify whether you want to scan for the first CNT or LNB or to start the scan at the current CNT or LNB. Use the LTERM name to narrow the scope of the scan. If the LTERM name is not used, clear the field.
	`CSSCCBLK = Current CNT or LNB address or zero`
	`CSSCNAME = LTERM name`

To scan	Initialize

RCNT	Specify whether you want to scan for the first RCNT or to start the scan at the current RCNT. Use the LTERM name to narrow the output of the scan. If the LTERM name is not used, clear the field.
	`CSSCCBLK = Current RCNT address or zero`
	`CSSCNAME = LTERM name`

CNT, LNB, or RCNT	Specify whether you want to scan for the first CNT, LNB, or RCNT, or to start the scan at the current CNT, LNB, or RCNT. Use the LTERM name to narrow the output of the scan. If the LTERM name is not used, clear the field.
	`CSSCCBLK = Current CNT, LNB, or RCNT address,`
	`or zero`
	`CSSCNAME = LTERM name`

SPQB	Specify whether you want to scan for the first SPQB or to start the scan at the current SPQB. Specify the USER name to narrow the output of the scan. If the USER name is not specified, clear the field.
	`CSSCCBLK = Current SPQB address or zero`
	`CSSCNAME = USER name`

VTCB	Specify whether you want to scan for the first VTCB or to start the scan at the current CLB. Specify either the NODE name alone, or the NODE and USER name to narrow the output of the scan. If the name fields are not specified, clear the fields.
	`CSSCCBLK = Current CLB address or zero`
	`CSSCNODE = NODE name`
	`CSSCUSER = USER name`

LOGON Descriptor	Specify whether you want to scan for the first LOGON descriptor or to start the scan at the current LOGON descriptor.
	`CSSCCBLK = Current LOGON descriptor address`
	`or zero`

USER Descriptor	Specify whether you want to scan for the first USER descriptor or to start the scan at the current USRD. `CSSCCBLK = Current USRD address or zero`
LLB	Specify whether you want to scan the first MSLINK or start the scan at the current MSLINK. `CSSCCBLK = Current LLB address or zero`
LCB	Specify whether you want to scan the first MSPLINK or start the scan at the current MSPLINK. `CSSCCBLK = Current LCB address or zero`

Output returned from SCAN Control Block Services

Depending on the type of scan specified, one of the following is returned in the CSSCNBLK field in the function-specific parameter list:

Search type	Output from service
SCAN CCB	Next CCB address
SCAN CNT or LNB	Next CNT or LNB address
SCAN RCNT	Next RCNT address
SCAN CNT, LNB, or RCNT	Next CNT, LNB, or RCNT address
SCAN SPQB	Next SPQB address
SCAN VTCB	Next CLB address
SCAN LOGON descriptor	CLB address of next LGND
SCAN USER descriptor	Next USRD address
SCAN LLB	Next LLB address
SCAN LCB	Next LCB address

IMS Callable AOI Services

DFSCAOI is the function-specific parameter list used for IMS Callable AOI Service requests. The DFSCAOI macro defines these requests.

The function-specific parameter list contains the information that AOI services needs to perform the function you requested (insert, enqueue, or cancel a message). The function-specific parameter list is also used to return data to your exit routine.

You must initialize this function-specific parameter list before calling DFSCSIF0 to activate AOI callable services. All fields that are not used as input to DFSCSIF0 should be cleared.

INSERT function

The INSERT function inserts the first, or a subsequent, message segment into a message buffer. The message segments are not available to the AO application until an enqueue is issued specifying an AOI token.

To request the INSERT function, initialize the following fields in the function-specific parameter list:

Field	Offset	Length	Description
CAOIFUNC	0	4	IMS AOI service function code value: 1 = INSERT message segment
CAOIDMTK	4	4	Directed message token
CAINMSEG	8	4	Address of message segment

ENQUEUE function

The ENQUEUE function inserts the last or only message segment into the message buffer, enqueues this message segment to the AOI token the requester has specified, and then makes the entire message available to the AO application.

If ENQUEUE is requested with a message segment address of 0, all previously inserted message segments are made available to the AO application.

To request the ENQUEUE function, initialize the following fields in the function-specific parameter list (DFSCAOI):

Field	Offset	Length	Description
CAOIFUNC	0	4	IMS AOI service function code value: 2 = ENQUEUE segment to AOI token
CAOIDMTK	4	4	Directed message token
CAENMSEG	8	4	Address of message buffer
CAENTCNT	12	4	Number of AOI token names in the token list addressed by the next word in this parameter list
CAENTLST	16	4	Address of a token list. Each 12-byte entry in the list contains the following: **Offset** **Description** **+0** The 8-byte alphanumeric AOI token name to which the message is to be enqueued **+8** The 4-byte code from the ENQUEUE function indicating whether the message was successfully enqueued to the AOI token. Possible codes are: **Code** **Meaning** **0** Enqueue to AOI token was successful. **1** Enqueue was unsuccessful. AOI token name was blanks. **2** Enqueue was unsuccessful. AOI token name contained invalid characters. **3** Enqueue was unsuccessful. Enqueue could not get AOIP storage. **4** Enqueue was unsuccessful. An internal latch error occurred.

CANCEL function

The CANCEL function cancels messages that have been inserted into the message buffer but not yet enqueued to the AOI token. Canceled messages are not made available to the application program.

To request the CANCEL function, initialize the following fields:

Field	Offset	Length	Description
CAOIFUNC	0	4	IMS AOI service function code value: 3 = CANCEL message segments
CAOIDMTK	4	4	Directed message token

Callable services return and reason codes

IMS callable services provides return and reason codes that describe why a callable service request did not complete successfully.

Callable services return and reason codes provide reasons for why function-specific parameter list, interface, and service processing errors occurred. These codes are in hexadecimal format.

Return codes (CSPLRTRN)

Return codes in field CSPLRTPN indicate why the request did not complete successfully.

Return codes are in field CSPLRTPN in the callable services parameter list. Following are the return codes indicating why the request did not complete successfully:

Return code	Meaning
X'04'	A callable service interface error occurred. The service request was not processed.
X'08'	Function-specific parameter list error. While processing the callable service request, an error occurred in the function-specific parameter list.
X'20'	Service processing error. An error occurred while processing the callable service request. The error could be a user error or an internal system error.

Callable service interface reason codes (CSPLRESN)

When the return code in the field CSPLRTRN is X'04', callable service interface reason codes in the field CSPLRESN explain why a callable service interface error occurred.

Following are the callable service interface reason codes:

Reason code	Meaning
X'04'	Callable services token is 0. The field CSPLTOKN in the callable services parameter list DFSCSPL is 0.
X'08'	Callable services token is invalid. The field CSPLTOKN in the callable services parameter list DFSCSPL does not contain a valid callable services token.
X'0C'	Service code is not specified. The field CSPLSERV in the callable services parameter list DFSCSPL is 0.
X'10'	Service code is invalid. The field CSPLSERV in the callable services parameter list DFSCSPL does not contain a valid callable service code. The service code is too large.
X'14'	Service is not supported. The field CSPLSERV in the callable services parameter list DFSCSPL contains a value for a callable service code that is not supported in the current environment or is a reserved function.
X'20'	The callable service token is wrong or the user exit is not eligible to invoke the AOI or control block callable service. If the callable service token is wrong, it belongs to another ITASK. Make sure that you have initialized the callable service token by calling DFSCSII0, before invoking a callable service by calling DFSCSIF0. Callable service tokens are valid only until the user exit returns control. Do not save and reuse a callable service token across multiple calls to an exit.
X'30'	Function code is not specified. The function code field in the function-specific parameter list is 0.
X'34'	Function code is invalid. The function code field in the function-specific parameter list contains a function code that is too large.

Reason code	Meaning
X'38'	Function is not supported. The function code field in the function-specific parameter list contains a value for a callable service function code that is not supported in the current environment or is a reserved function.

Function-specific parameter list reason codes (CSPLRESN)

When the return code in the field CSPLRTRN is 8 or 20, an error occurred in the function-specific parameter list. The function-specific parameter list reason codes are stored in the field CSPLRESN and are described by service and by function.

GET storage service reason codes

When an error occurs in the GET storage service function-specific parameter list, the return code in the field CSPLRTRN is 8 or 20. The reason codes are stored in the field CSPLRESN and are described by service and by function.

Following are the reason codes for GET function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Invalid subpool parameter. The field CSGTSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
X'8'	Invalid location parameter. The field CSGTLOC in the function-specific parameter list DFSCSTRG contains an invalid storage location value.
X'C'	Invalid boundary parameter. The next CSGTBNDY in the function-specific parameter list DFSCSTRG contains an invalid storage boundary value.
X'10'	Length parameter not specified. The field CSGTLEN in the function-specific parameter list DFSCSTRG is 0.

When CSPLRTRN = 20

If you receive any reason code not listed in the following table, contact IBM Software Support.

Reason code	Meaning
X'06 00 00 04'	Storage could not be allocated.
X'06 00 00 08'	Parameter list error.

FREE storage service reason codes

When an error occurs in the FREE storage service function-specific parameter list, the return code in the field CSPLRTRN is 8 or 20. The reason codes are stored in the field CSPLRESN and are described by service and by function.

Following are the reason codes for FREE function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Invalid subpool parameter. The field CSFRSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.

Reason code	Meaning
X'8'	Address parameter not specified. The field CSFRSTAD in the function-specific parameter list DFSCSTRG is 0.
X'C'	Length parameter not specified. The field CSFRLEN in the function-specific parameter list DFSCSTRG is 0.

When CSPLRTRN = 20

If you receive any reason code not listed in the following table, contact IBM Software Support.

Reason code	Meaning
X'07 00 00 04'	Storage was not released. A value in the second byte of the reason code is provided by the associated z/OS Service. For example, the 04 in the second byte of reason code 07 04 00 04 is returned from z/OS FREEMAIN. Additional information can be found in the IMODULE FREESTOR Return Codes section of *IMS Version 15.2 Messages and Codes, Volume 4: IMS Component Codes*.
X'07 00 00 08'	Parameter list error.
X'07 00 00 0C'	Unable to locate storage descriptor block. Storage address might be invalid or storage subpool specification might be incorrect.

LOAD storage service reason codes

When an error occurs in the LOAD storage service function-specific parameter list, the return code in the field CSPLRTRN is 8 or 20. The reason codes are stored in the field CSPLRESN and are described by service and by function.

Following are the reason codes for LOAD function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Invalid subpool parameter. The field CSLDSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
X'8'	Invalid location parameter. The field CSLDLOC in the function-specific parameter list DFSCSTRG contains an invalid module location value.
X'C'	Invalid use parameter. The field CSLDUSE in the function-specific parameter list DFSCSTRG contains an invalid module reuse value.
X'10'	Name parameter not specified. The field CSLDNAME in the function-specific parameter list DFSCSTRG does not contain a module name.
X'14'	The caller is running in cross memory mode, and the primary address space is not CTL or DLI.

When CSPLRTRN = 20

If you receive any reason code not listed in the following table, contact IBM Software Support.

Reason code	Meaning
X'02 00 00 04'	Module was not found.
X'02 00 00 08'	DFSMODU0 allocation error.
X'02 00 00 0C'	BLDL/FETCH allocation error.

Reason code	Meaning
X'02 00 00 10'	FETCH/BLDL I/O error occurred loading the requested module.
X'02 00 00 24'	DCB was not open for BLDL.
X'02 00 00 28'	Caller was authorized, but module was found in unauthorized library.

DELETE storage service reason codes

When an error occurs in the DELETE storage service function-specific parameter list, the return code in the field CSPLRTRN is X'8' or X'20'. The reason codes are stored in the field CSPLRESN and are described by service and by function.

Following are the reason codes for DELETE function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Invalid subpool parameter. The field CSDLSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
X'8'	Name and address was not specified. The field CSDLNAME in the function-specific parameter list DFSCSTRG does not contain a module name, and CSDLEP does not contain a module address.
X'C'	The caller is running in cross memory mode, and the primary address space is not CTL or DLI.

When CSPLRTRN = 20

If you receive any reason code not listed in the following table, contact IBM Software Support.

Reason code	Meaning
X'04 00 00 04'	Module was not found.
X'04 00 00 0C'	Module storage was not released.

FIND control block service reason codes

When an error occurs in the FIND control block service function-specific parameter list, the return code in the field CSPLRTRN is 8 or 20.

Following are the reason codes for FIND function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	FIND type was not specified. The field CSFDTYPE in the function-specific parameter list DFSCCBLK is 0.
X'8'	FIND type was invalid. The field CSFDTYPE in the function-specific parameter list DFSCCBLK does not contain a valid control block search type value. The search type value is too large.
X'C'	CCBID was not specified. The field CSFDEIB in the function-specific parameter list DFSCSTRG does not contain an EBCDIC CCB identifier, and CSFDBID does not contain a binary CCB identifier.

Reason code	Meaning
X'10'	Control block name was not specified. The field CSFDNAME in the function-specific parameter list DFSCCBLK does not contain a name.

When CSPLRTRN = 20

Following are the reason codes you might get when searching CCB, CNT, LNB, RCNT, SPQB, CNT, descriptor and USER descriptor control block types:

Reason code	Meaning
X'4'	Block was not found.
X'40 00 00 00'	CBTS latch held, cannot process request.

Following are the reason codes you might get when searching VTCB and LOGON descriptor control block types:

Reason code	Meaning
X'4'	Cannot find CLB with VTAM CID or node/descriptor name.
X'8'	NO VTCBs/LGNDs are in system.
X'40 00 00 00'	CBTS latch held, cannot process request.

The following are the reason codes that can be encountered when searching for a transaction control block type.

Reason code	Meaning
X'8'	Transaction was not found.
X'40 00 00 00'	CBTS latch held, cannot process request.

SCAN control block service reason codes

When an error occurs in the SCAN control block service function-specific parameter list, the return code in the field CSPLRTRN is 8 or 20.

Following are the reason codes for SCAN function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	SCAN type was not specified. The field CSSTYPE in the function-specific parameter list DFSCCBLK is 0.
X'8'	SCAN type was invalid. The field CSSCTYPE in the function-specific parameter list DFSCCBLK does not contain a valid control block search type value. The search type value is too large or is a reserved function.

When CSPLRTRN = 20

Following are the reason codes you might get when searching CCB, CNT, LNB, RCNT, SPQB, and USER descriptor control block types:

Reason code	Meaning
X'4'	End of queue was found.
X'8'	No block is in system.
X'14'	Bad INUSE call. Verify that the CSSCCBLK and CSSCNAME fields are properly initialized.
X'18'	Bad NOUSE call. Verify that the CSSCCBLK and CSSCNAME fields are properly initialized.
X'40 00 00 00'	CBTS latch held, cannot process request.

Following are the reason codes you might get when searching VTCB and LOGON descriptor control block types:

Reason code	Meaning
X'4'	Cannot find VTCB matching arguments.
X'8'	No VTAM nodes were in system.
X'40 00 00 00'	CBTS latch held, cannot process request.

INSERT AOI service reason codes

When an error occurs in the INSERT AOI service function-specific parameter list, the return code in the field CSPLRTRN is 8 or 20.

Following are the reason codes for INSERT function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Directed message token was 0.
X'8'	Directed message token was invalid.
X'C'	Message segment address was 0.
X'10'	Message segment length (LL field) was 0.

When CSPLRTRN = 20

Reason code	Meaning
X'4'	IMS could not get the storage required to process the call.

ENQUEUE AOI service reason codes

When an error occurs in the ENQUEUE AOI service function-specific parameter list, the return code in the field CSPLRTRN is 8.

Following are the reason codes for ENQUEUE function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Directed message token was 0.
X'8'	Directed message token was invalid.

Reason code	Meaning
X'10'	Message segment address was specified, but segment length (LL field) was 0.
X'14'	AOI token count field was 0.
X'18'	AOI list token address was 0.
X'1C'	One or more tokens was processed successfully.
X'20'	No tokens were processed successfully.

CANCEL AOI Service reason codes

When an error occurs in the CANCEL AOI Service function-specific parameter list, the return code in the field CSPLRTRN is 8.

Following are the reason codes for CANCEL function parameter errors:

When CSPLRTRN = 8

Reason code	Meaning
X'4'	Directed message token was 0.
X'8'	Directed message token was invalid.
X'C'	No message exists to cancel.

Callable services request example

An exit routine could use IMS callable services using DFSCSII0.

The following example depicts how an exit routine could use IMS callable services. In the example, the storage returned from DFSCSII0 is divided into three areas. These areas are for the parameter lists used for the call to DFSCSIF0. The first area is used for the z/OS CALL parameter list, the second for the IMS callable service parameter list, and the third for the function specific parameter list. The labels, CSICLLEN and CSPLPLEN, used in the examples are defined as EQU statements in the macro DFSCSIPL. These labels represent the length of the z/OS parameter list built by the CALL macro and the length of the IMS callable services parameter list.

```
*********************************************************************
*                                                                   *
*       --------------------------------                            *
*       GETSTOR - GET STORAGE SUBROUTINE                            *
*       --------------------------------                            *
*                                                                   *
*       THIS SUBROUTINE INVOKES IMS callable services TO            *
*       GET WORKING STORAGE.  THE CALLER PASSES THE REQUIRED        *
*       STORAGE LENGTH.  THE SUBROUTINE THEN OBTAINS PRIVATE,       *
*       31-BIT STORAGE ON A DOUBLEWORD BOUNDARY.                    *
*                                                                   *
*                                                                   *
*       INPUT REGISTERS:                                            *
*          R8  = REQUESTED STORAGE LENGTH                           *
*          R9  = ECB ADDRESS                                        *
*          R10 = LINKAGE REGISTER                                   *
*                CALLED BY BAL 10,GETSTOR                           *
*                                                                   *
*       OUTPUT REGISTERS:                                           *
*          R1  = STORAGE ADDRESS                                    *
*          R9  = ECB ADDRESS                                        *
*          R10 = LINKAGE REGISTER                                   *
*          R15 = RETURN CODE                                        *
*                0         - CALL COMPLETED SUCCESSFULLY            *
*                NON-ZERO  - STORAGE REQUEST FAILED                 *
*                            RETURN CODE FROM IMS CALLABLE STORAGE  *
*                            SERVICES - GET STORAGE FUNCTION        *
*                                                                   *
*       REGISTER USAGE:                                             *
```

```
*          R0  = WORK REGISTER                                        *
*          R1  = WORK REGISTER                                        *
*          R2  = IMS CALLABLE SERVICE TOKEN                           *
*          R3  = IMS callable services PARAMETER LIST                 *
*          R4  = IMS STORAGE SERVICES PARAMETER LIST                  *
*          R5  = z/OS CALL PARAMETER LIST                              *
*          R8  = REQUESTED STORAGE LENGTH                             *
*          R9  = ECB ADDRESS                                          *
*          R14 = WORK REGISTER                                        *
*          R15 = WORK REGISTER                                        *
*                                                                     *
***********************************************************************
GETSTOR  DS    0H
         SPACE
***********************************************************************
*       INVOKE CALLABLE SERVICES INITIALIZATION ENTRY POINT           *
*       DFSCSII0, TO OBTAIN THE CALLABLE SERVICE TOKEN AND            *
*       PARAMETER LIST STORAGE.                                       *
***********************************************************************
         LR    1,9                ECB ADDRESS
         CALL  DFSCSII0           INVOKE INIT ENTRY POINT
         LTR   15,15              CALL SUCCESSFUL?
         BNZ   GSTREXIT           NO, ERROR RETURN
         SPACE
***********************************************************************
*       R1 CONTAINS A PARAMETER LIST ADDRESS.                         *
*       OFFSET 0 IN THE LIST CONTAINS THE 4-BYTE CALLABLE             *
*       SERVICE TOKEN.  EXTRACT THE TOKEN FROM THE PARAMETER          *
*       LIST FOR USE ON THE GET STORAGE REQUEST.                      *
***********************************************************************
         LR    5,1                COPY STORAGE ADDRESS
         L     2,0(,5)            CALLABLE SERVICE TOKEN
         SPACE
***********************************************************************
*       R5 CONTAINS THE ADDRESS TO USE FOR THE PARAMETER              *
*       LIST FOR THE z/OS CALL MACRO.  USING THE EQU LABELS           *
*       IN MACRO DFSCSIPL, CARVE THE STORAGE RETURNED BY              *
*       DFSCSII0 INTO SEPARATE PARAMETER LISTS TO BE USED             *
*       ON THE CALL TO DFSCSIF0.                                      *
***********************************************************************
         LA    3,CSICLLEN(,5)     CALLABLE SERVICE PARM LIST ADDR
         LA    4,CSPLPLEN(,3)     STORAGE SERVICES PARM LIST ADDR
         SPACE
***********************************************************************
*       PARAMETER LIST RETURNED FROM DFSCSII0 HAS BEEN CARVED INTO    *
*       THREE PARTS:                                                  *
*                                                                     *
*       R5               R3                   R4                    *
*        ---------------------------------------------------          *
*       |  Z/OS CALL AREA  |  IMS CALL SVC AREA  |  STG SVC AREA  |    *
*        ---------------------------------------------------          *
*                                                                     *
***********************************************************************
         SPACE
***********************************************************************
*       INITIALIZE CALLABLE SERVICE PARAMETER LIST.                   *
*                                                                     *
*       ENTIRE LIST IS CLEARED SO ALL RESERVED AND NON-INPUT          *
*       FIELDS (SUCH AS THE RETURN AND REASON CODES)                  *
*       ARE SET TO ZERO.  THE CALLABLE SERVICE CODE IS                *
*       INITIALIZED TO REQUEST STORAGE SERVICES                       *
*       AND THE CALLABLE SERVICE TOKEN IS SAVED IN THE LIST.          *
***********************************************************************
         USING CSPARMS,3          CALLABLE SERVICES PARM LIST DSECT
         XC    CSPARMS(CSPLPLEN),CSPARMS  CLEAR CALLABLE SERVICES LIST
         LA    0,CSPLSTRG         STORAGE SERVICE CODE
         ST    0,CSPLSERV         INSERT SERVICE CODE IN LIST
         ST    2,CSPLTOKN         INSERT CALLABLE SERVICE TOKEN
         SPACE
***********************************************************************
*       INITIALIZE STORAGE SERVICE PARAMETER LIST                     *
*                                                                     *
*       ENTIRE LIST IS CLEARED SO ALL RESERVED AND NON-INPUT          *
*       FIELDS (SUCH AS THE RETURN AND REASON CODES)                  *
*       ARE SET TO ZERO.  THE STORAGE SERVICES                        *
*       FUNCTION CODE IS INITIALIZED TO REQUEST THE GET STORAGE       *
*       FUNCTION.  PARAMETERS ARE INITIALIZED TO OBTAIN 31-BIT,       *
*       PRIVATE STORAGE IN SUBPOOL 0 ON A DOUBLEWORD BOUNDARY.        *
*                                                                     *
***********************************************************************
         USING CSSTRG,4           STORAGE SERVICES PARM LIST DSECT
         XC    CSSTRG(CSGTPLEN),CSSTRG  CLEAR STORAGE SERVICES LIST
```

```
         LA    0,CSSTGET           GET STORAGE FUNCTION CODE
         ST    0,CSSTFUNC          INIT FUNCTION CODE PARAMETER
         SPACE
         ST    8,CSGTLEN           INIT STORAGE LENGTH PARAMETER
         SPACE
         LA    0,CSGTPRI           PRIVATE STORAGE INDICATOR
         ST    0,CSGTSP            INIT STORAGE SUBPOOL INDICATOR
         SPACE
         LA    0,CSGT31B           31-BIT STORAGE INDICATOR
         ST    0,CSGTLOC           INIT STORAGE LOCATION PARAMETER
         SPACE
         LA    0,CSGTDBLW          DOUBLE WORD BOUNDARY INDICATOR
         ST    0,CSGTBNDY          INIT STORAGE BOUNDARY PARAMETER
         SPACE
*********************************************************************
*       THE CALLABLE SERVICES PARAMETER LIST HAS BEEN INITIALIZED   *
*       TO INVOKE IMS STORAGE SERVICES.  THE STORAGE SERVICES       *
*       PARAMETER LIST HAS BEEN INITIALIZED TO OBTAIN USER STORAGE. *
*       ISSUE THE IMS CALLABLE SERVICE REQUEST TO OBTAIN STORAGE.   *
*********************************************************************
         CALL  DFSCSIF0,((3),(4)),MF=(E,(5))
         LTR   15,15               STORAGE REQUEST SUCCESSFUL?
         BNZ   GSTREXIT            NO, RETURN TO CALLER
         SPACE
         L     1,CSGTADDR          STORAGE ADDRESS
         SPACE
*********************************************************************
*       RETURN TO CALLER                                            *
*********************************************************************
GSTREXIT DS    0H
         BR    10                  RETURN TO CALLER
         LTORG
         DFSCSIPL
```

Control block usage

Review this directory of the control blocks, their associated fields that are intended for access by exit routines, and restrictions of their use.

If only certain fields within a control block are intended for your use, they are listed next to the control block name in the following table. If a field does not appear next to the control block name, it is not intended for your use. Unless otherwise specified, the only information that is part of the interface for exit routines is the control block name and any specific fields associated with that control block. For a field that is part of the interface, the only information that is part of the interface for exit routines is the named field.

The following control blocks and their associated fields and flags, shown in the following table, are intended for use as, or as part of, a product-sensitive interface. Flags are enclosed in parenthesis next to their associated fields.

Table 7. Control blocks and associated fields and flags

Control block name	Fields and flags intended for use
CCB	CCBNUMB
CIB	CIBMNAME, CIBDTYP (CIBDNDS)
CLB	CLBNAME, CLBCURR, CLBCNTQB
CNT, LNB	CNTDEQCT, CNTENQCT, CNTNAME, CNTDQCT, CNTCTBPT, CNTCNTPT
CTB	CTBCTT, CTBTERM, CTBFLAG1 (CTB1SIGN, CTB1PRES), CTBFLAG2 (CTB2LOCK, CTB2TEST, CTB2EXCL), CTBFLAG3 (CTB3SEG1), CTBACTL (CTBAEOM, CTBAINC), CTBFEAT, CTBINCT, CTBOUTCT, CTBCNT, CTBCIBPT, CTBPRSTN, CTBCNTPT, CTBFLAG6 (CTB6SDON, CTB6TRNI), CTBUSID, CTBOUSID
CTT	CTTDEVIC (CTTD3286, CTTDTYP1, CTTDLU4), CTTSEND, CTTEDIT, CTTIEDIT, CTTOPT2 (CTT2DIT), CTTOPT5 (CTT5DYN)

Table 7. Control blocks and associated fields and flags (continued)

Control block name	Fields and flags intended for use
CVB	CVBCCMD
DFSPDA	PDAPDE, PDANUM, PDADORG, PDALSTRL, PDAUSR1, PDAUSR2, PDAUSR3, PDAUSR4, PDAUSR5, PDAPLEN
DFSPDAE	PDAPN, PDASTRG, PDAPID, PDARAP, PDABLKR, PDASTRGL, PDAFLAG1 (PDAF101), PDAELEN
DFSPECA	PECDBN, PECRC, PECFDB, PECFDB2, PECKEY, PECCPID, PECKEYL, PECACT, PECFLAG1 (PEC1NEWP), PECFLAG2, PECUSER
FEIB	FEIBOFLG (FEIBRPQ1, FEIBERP, FEIBTMED), FEIBMSGN, FEIBLTRM, FEIBMSG, FEIBUNID, FEIBNDST, FEIBERPN, FEIBLDST, FEIBULNG FEIBUSER, FEIBIMID
MFSFLDE	FLDFLAG (FLDOPT, FLDEXIT, FLDATTR, FLDEATR), FLDELTH, FLDVECT, FLDLTH, FLDADDR (OPT3LTH, OPT3ID, OPT3DATA)
MFSSEGE	SEGFLAG, SEGOPT (SEGEXIT, SEGECHO), SEGVECT, SEGLTH, SEGFLDRC (SEGDL)
MSNB	MSNFLG1 (MSN1DEQ), MSNFLG3 (MSN3DQND, MSN3DQLM)
PDIR	PDIRSYM, PDIRCODE (PDIRLOCK, PDIRNOSC, PDIRSCHD, PDIRDBST, PDIRBALG), PDIROPTC (PDIRRETN, PDIRGPSB, PDIRDOPT, PDIRPARL, PDIRBAD), PDIRFLG3 (PDIRIFPR, PDIRIFPM, PDIRIFPU)
RCNT	CNTDEQCT, CNTENQCT, CNTNAME, CNTDQCT
SCD	SSCDIMID, SCDQTU, SCDQTL, SCDSSTYP (SCDSSDBC, SCDSSDCC), SSCDIMSR, SSCDIMSL
SMB	SMBDEQCT, SMBENQCT, SMBTRNCD, SMBSTATS (SMBSRESP, SMBSMULT, SMBSNOQU, SMBNOSC, SMBLOCK, SMBSQERR), SMBFLAG1 (SMB1CONV, SMB1UPP, SMBCPIC, SMB1NORE, SMB1INIT), SMBFLAG2 (SMB2DRRT, SMBFPPTX, SMBFPXCL, SMB2SMS, SMB2RMT), SMBFLAG3 (SMBBAD, SMB3WFI), SMBFLAG5 (SMBINQN, SMB5TLS), SMBPRIOR, SMBCLASS, SMBSPAL, SMBLMTCT, SMBCOUNT, SMBSIDR, SMBSIDL, SMBMXRGN, SMBPARLM, SMBAOIFL (SMBTCMDA, SMBNOSCH), SMBPDIRN, SMBRCTEN
SPQB, USRD	SPQBHSQN

The following table provides a list, by exit, of the control blocks that are intended for use as, or as part of, a product-sensitive interface:

Table 8. Exit routines and associated control blocks

Exit name or type	Associated control blocks
DBFHAGU0	SCD
DBFHDC40	none
DBFHDC44	none
DBFUMSE1	none
DBFLHSH0	none
AOIE	none
DFSAOUE0	CLB, CTB, SCD

Table 8. Exit routines and associated control blocks (continued)

Exit name or type	Associated control blocks
BSEX	none
DFSCCMD0	CLB, CTB, CTT, CVB, SCD
DFSCKWD0	none
DFSCMPX0	none
DFSCMTU0	none
DFSCMUX0	MSNB
DFSCNTE0	CLB, CNT, CTB
DFSCONE0	CCB, CTB, PDIR, SCD, SPQB, SMB
DFSCSGN0	CTB, SCD
DFSCSMB0	CLB, CTB
DFSCTRN0	CLB, CNT, CTB, PDIR, SCD, SMB
DFSCTSE0	CNT, CTB, PDIR, SCD, SMB
DFSCTTO0	CLB, CNT, CTB, SCD
DFSFDOT0	none
DFSFEBJ0	FEIB, PDIR, SMB
LOGWRT	none
DFSFTFX0	none
DFSGMSG0	none
DFSGPIX0	PDIR, SMB
DFSHDC40	DMBDACS
DFSINSX0	CLB, SCD
DFSINTX0	CLB, SCD
DFSI7770	CLB, CNT, CTB, SCD
DFSLGFX0	CLB, SCD
DFSLGNX0	CLB, SCD
DFSLUEE0	none
DFSME000	MFSFLDE
DFSME127	MFSSEGE, CLB
DFSMSCE0	SCD
NDMX	none
DFSO7770	CLB, CTB, CTT, SCD
DFSPIXT0	CTB, PDIR, SMB
PPUE	none
DFSPRE60	none

Table 8. Exit routines and associated control blocks (continued)

Exit name or type	Associated control blocks
DFSPRE70	none
DFSPSE00	DFSPECA, DFSPDA, DFSPDAE
DFSQSPC0	PDIR, SCD, SMB
DFSSBUX0	none
DFSSGFX0	CLB, SCD
DFSSGNX0	CIB, CLB, CTB, CTT, SCD
DFSSIML0	CLB, CNT, CTB, CTT, SCD
DFSS7770	CLB, CNT, CTB, CTT, SCD
DFSTXIT0	none
DFSYORU0	none
OTMAIOED	none
OTMAYPRX	none
DFS29800	CLB, CNT, CTB, PDIR, SCD, SMB
DFS36010	CLB, CTB, SCD
DSPCEXT0	none

Customization exit routines

IMS provides sample exit routines and programs for most exit points.

The location of the sample exit routines and programs are listed in the following table.

Table 9. Exit routines and their location

Exit routine or user exit type	Location	Description
BSEX	No sample	Build Security Environment exit routine
DBFHAGU0	IMS.SDFSSRC	IMS Fast Path Sample User Input Exit
DBFHDC40 / DBFHDC44	IMS.SDFSSRC	IMS/FP Randomizing Exit
DBFLHSH0	IMS.SDFSSRC	Data Entry Database Resource Name hash routine
DBFUMSE1	Sample provided in IBM Documentation	DEDB Sequential Dependent Scan utility exit routine
DFSAOE00	IMS.SDFSSMPL	Type-2 Automated Operator exit (AOIE) routine sample
DFSAOUE0	IMS.SDFSSMPL	AOI User exit routine sample program
DFSBXITA	IMS.SDFSSMPL	CEEBXITA Assembler user exit routine for IMS

Table 9. Exit routines and their location (continued)

Exit routine or user exit type	Location	Description
DFSCCMD0	IMS.SDFSSMPL	Command Authorization user exit routine sample
DFSCKWD0	IMS.SDFSSRC	Command Keyword Table
DFSCMPX0	IMS.SDFSSMPL	User-data Compression program
DFSCMTU0	No sample	User Message Table
DFSCMUX0	IMS.SDFSSRC	Message Control/Error exit routine
DFSCNTE0	IMS.SDFSSMPL	Sample CNT Destination edit routine
DFSCONE0	IMS.SDFSSMPL	Conversational user exit routine
DFSCSGN0	IMS.SDFSSMPL	COMM / SIGN exit routine sample
DFSCSMB0	IMS.SDFSSMPL	Transaction Code (Input) edit routine
DFSCTRN0	IMS.SDFSSMPL	COMM Transaction Authorization exit routine sample
DFSCQEX0	IMS.SDFSSMPL	IMS CQS structure event user exit (ICQSSTEV)
DFSCSTX0	IMS.SDFSSMPL	IMS CQS event user exit (ICQSEVNT)
DFSCTSE0	No sample	Security Reverification exit routine
DFSCTTO0	IMS.SDFSSMPL	Sample PTERM (Output) edit routine
DFSFDOT0	IMS.SDFSSMPL	IMS Dump Override table
DFSFEBJ0	IMS.SDFSSMPL	Front End Switch user exit routine
DFSFIDN0	IMS.SDFSSMPL	ESAF In-Doubt Notification exit routine
DFSFTFX0	IMS.SDFSSRC	Log Filter exit routine
DFSGMSG0	IMS.SDFSSMPL	Greeting Messages user exit routine
DFSGPIX0	No sample	Global Physical Terminal (Input) edit routine
DFSHDC40	IMS.SDFSSRC	HDAM and PHDAM randomizing routine
DFSINSX0	IMS.SDFSSMPL	Output Destination Creation user exit routine
DFSINTX0	IMS.SDFSSMPL	IMS Initialization user exit routine
DFSITRX0	IMS.SDFSSMPL	IMS Initialization and Termination user exit (INITTERM)
DFSKMPX0	IMS.SDFSSMPL	User Data Compression program
DFSLGFX0	IMS.SDFSSMPL	IMS Logoff user exit routine
DFSLGNX0	IMS.SDFSSMPL	User Logon exit routine
DFSLUEE0	IMS.SDFSSRC	LU 6.2 Edit exit routine

Table 9. Exit routines and their location (continued)

Exit routine or user exit type	Location	Description
DFSME000	IMS.SDFSSRC	Input Message Field edit routine
DFSME127	IMS.SDFSSRC	Input Message Segment edit routine
DFSMONX0	IMS.SDFSSMPL	IMS Monitor (IMSMON) sample user exit routine
DFSMSCE0	IMS.SDFSSMPL	TM and MSC Message Routing and Control user exit routine
DFSPIXT0	IMS.SDFSSMPL	Physical Termination Input Edit routine sample
DFSPPUE0	No sample	Partner Product exit routine
DFSPRE60	IMS.SDFSSMPL	System Definition Preprocessor exit routine (input phase)
DFSPRE70	IMS.SDFSSMPL	System Definition Preprocessor exit routine (name check complete)
DFSPSE00	IMS.SDFSSMPL	Sample Partition Selection exit routine
DFSQSPC0	IMS.SDFSSRC	Queue Space Notification exit routine
DFSREXXU	IMS.SDFSSMPL	REXXTDLI Sample user exit routine
DFSSBUX0	No sample	Sequential Buffering Initialization exit routine
DFSSGFX0	IMS.SDFSSMPL	Sign-off user exit routine
DFSSGNX0	IMS.SDFSSMPL	Sign-on user exit routine example
DFSSIML0	IMS.SDFSSMPL	Shared Printer exit routine
DFSTXIT0	IMS.SDFSSRC	Time-Controlled Operations exit routine
DFSUSO	IMS.SDFSSMPL	Sample z/OS IEFUSO SYSOUT limits exit routine
DFSUTL	IMS.SDFSSMPL	Sample MVS™ IEFUTL Timeout exit routine
DFSX09B	IMS.SDFSSMPL	Sample z/OS JES2 HASX09B Output excession options exit routine
DFSYDRU0	IMS.SDFSSMPL	OTMA User Data Formatting exit
DFS29800	No sample	2972/2980 Input Edit Routine
DFS36010	IMS.SDFSSMPL	COMM DEV MOD (3600), Sample 3601 Input edit routine
DSPBUFFS	IMS.SDFSSRC	Buffer Size Specification facility
DSPCEXT0	IMS.SDFSSMPL (sample is named DSPCEXT1)	RECON I/O exit routine
DSPDCAX0	IMS.SDFSSMPL	Sample DBRC SCI Registration exit routine

Table 9. Exit routines and their location (continued)

Exit routine or user exit type	Location	Description
DSPSCIX0	IMS.SDFSSMPL	Sample DBRC SCI Registration exit routine
LOGWRT	No sample	Logger exit routine
NDMX	IMS.SDFSSMPL	Non-Discardable Messages (NDMX) user exit
OTMAIOED	IMS.SDFSSMPL	OTMA Input/Output Edit user exit
OTMARTUX	IMS.SDFSSMPL	OTMA Resume TPIPE Security exit routine
OTMAYPRX	IMS.SDFSSMPL	OTMA Destination Resolution exit routine
PGMCREAT	IMS.SDFSSMPL	PGMCREAT User Exit
RASE	IMS.SDFSSMPL	Resource Access Security exit routine sample

IMS.SDFSSMPL data set

The IMS.SDFSSMPL data set contains source code modules that you can customize for various purposes.

Table 10. IMS.SDFSSMPL data set exit routines and descriptions

Exit routines	Description
DBFMLBX0	Fast Path MADS I/O Timing user hash routine
DFSAOE00	Type-2 Automated Operator exit (AOIE) routine sample
DFSAOUE0	AOI User exit routine sample program
DFSBXITA	CEEBXITA Assembler user exit routine for IMS
DFSCCMD0	Command Authorization user exit routine sample
DFSCMPX0	User-data Compression program
DFSCNTE0	Sample CNT Destination edit routine
DFSCONE0	Conversational user exit routine
DFSCSGN0	COMM / SIGN exit routine sample
DFSCSMB0	Transaction Code (Input) edit routine
DFSCQEX0	IMS CQS structure event user exit (ICQSSTEV)
DFSCSTX0	IMS CQS event user exit (ICQSEVNT)
DFSCTRN0	COMM Transaction Authorization exit routine sample
DFSCTTO0	Sample PTERM (Output) edit routine
DFSFDOT0	IMS Dump Override table
DFSFEBJ0	Front End Switch user exit routine
DFSGMSG0	Greeting Messages user exit routine

Table 10. IMS.SDFSSMPL data set exit routines and descriptions (continued)

Exit routines	Description
DFSIDEF0	IMS Installation Defaults Block
DFSINSX0	Output Destination Creation user exit routine
DFSINTX0	IMS Initialization user exit routine
DFSITRX0	IMS Initialization and Termination user exit (INITTERM)
DFSKMPX0	User Data Compression program
DFSLGFX0	IMS Logoff user exit routine
DFSLGNX0	User Logon exit routine
DFSMSCE0	TM and MSC Message Routing and Control user exit routine
DFSMONX0	IMS Monitor (IMSMON) sample user exit routine
DFSNDMX0	Non-Discardable Messages (NDMX) user exit
DFSPIXT0	Physical Termination Input Edit routine sample
DFSPRE60	System Definition Preprocessor exit routine (input phase)
DFSPRE70	System Definition Preprocessor exit routine (name check complete)
DFSPSE00	Sample Partition Selection exit routine
DFSRAS00	Resource Access Security exit routine sample
DFSREXXU	REXXTDLI Sample user exit routine
DFSSGFX0	Sign-off user exit routine
DFSSGNX0	Sign-on user exit routine example
DFSSIML0	Shared Printer exit routine
DFSUSO	Sample z/OS IEFUSO SYSOUT limits exit routine
DFSUTL	Sample MVS IEFUTL Timeout exit routine
DFSX09B	Sample z/OS JES2 HASX09B Output excession options exit routine
DFSYCWAT	Sample program that suspends the currently executing task
DFSYDRU0	OTMA User Data Formatting exit
DFSYIOE0	OTMA Input/Output Edit user exit
DFSYPRX0	OTMA Destination Resolution exit routine
DFS36010	COMM DEV MOD (3600), Sample 3601 Input edit routine
DSPAPSMP	Example Program Using the DBRC API
DSPCEXT1	Sample DBRC I/O exit routine
DSPDCAX0	Sample DBRC Command Authorization user exit routine
DSPSCIX0	Sample DBRC SCI Registration exit routine
PGMCREAT	Dynamically create the runtime control block (PDIR) for an application program when the application program is scheduled by IMS.

Chapter 2. Database Manager exit routines

Use the database manager exit routines to initialize products that run with IMS, control operations related to subsystems, and enhance the maintenance and control of segments.

Batch application exit routine (DFSISVI0)

The batch application exit routine (DFSISVI0) routine is called immediately before linking to the batch application program. The exit routine has no predefined purpose. You can use it to allow the initialization of products that run with IMS. The exit is called prior to calling the application program.

Subsections:

About this routine

The Batch Application exit routine is applicable to IMS DB and IMS TM batch environments, and batch types DBB, DLI, and ULU. The exit routine is called if it is available in IMS.SDFSRESL.

You can link-edit the exit routine as needed, and will process in TASK mode. The exit routine's addressing mode can be either 24 or 31. It is given control in its defined AMODE and can return control to IMS in either 24- or 31-bit addressing mode.

Table 11. Batch application exit routine attributes

Attribute	Description
IMS environments	DB Batch, TM Batch.
Naming convention	Must be named DFSISVI0.
Link editing	After you compile your routine, include it into IMS.SDFSRESL or into any operating system-partitioned data set to which access is provided by using a JOBLIB or STEPLIB JCL statement.
Including the routine	No special steps required.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	No sample exit routine is provided.

Calling this routine

This exit routine is called using standard linkage conventions.

Communicating with IMS

IMS communicates with this routine through the entry registers, a parameter list, and the exit registers.

Content of Registers on Entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the exit parameter list.
13	Address of a single, standard save area.

Register	Content
14	Return address to IMS.
15	Entry point of this exit routine.

Parameter list

The following parameter list is provided to the exit routine:

00
Address of the application PCB list.

04
Address of PXPARMS

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains the return code. A return code of 12 indicates that the exit does not want IMS processing to continue.

Return code	Meaning
00	Continue normal IMS processing.
04	Undefined. Treated like a return code of 00.
08	Undefined. Treated like a return code of 00.
12	Terminate IMS processing with U0099 abend.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

IMS Catalog Definition exit routine (DFS3CDX0)

Use the IMS Catalog Definition exit routine (DFS3CDX0) to provide the settings and attributes of the IMS catalog to batch application programs. Using this exit routine is an alternative to referencing the DFSDF*xxx* member of the IMS.PROCLIB data set in the JCL of batch application programs.

This exit routine is available in batch processing environments only.

About this routine

Table 12. Catalog Definition exit routine attributes

Attribute	Description
IMS environments	IMS batch
Naming convention	Must be named DFS3CDX0

Table 12. Catalog Definition exit routine attributes (continued)

Attribute	Description

Binding	• You must bind this exit routine module into IMS.SDFSRESL or a concatenated library.
	• You must code this exit routine module as reentrant.
	• IMS batch processing attempts to load this exit routine, then attempts to load a DFSDF*xxx* member of the IMS.PROCLIB data set if this exit routine is not found.
	• If you enable the IMS catalog with this exit routine (function code 1), you must ensure that the catalog resource members (DFSCP000, DFSCD000, DFSCX000) have been added to the PSB and DBD libraries with the appropriate PSB generation or DBD generation utility.
	• If the IMS management of ACBs is indicated by X'80' in the DXPL_FUNCV2 field, the IMS management of ACBs must be enabled in the IMS system.
	The following example JCL shows how to bind the exit routine module into IMS.SDFSRESL.
	`//LINKIT JOB 1,MSGLEVEL=1`
	`//LINK EXEC PGM=IEWL,PARM=RENT`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL.,DISP=SHR`
	`//OBJIN DD DSN=IMS.USERLIB.,DISP=SHR`
	`//SYSLIN DD *`
	`INCLUDE OBJIN(DFS3CDX0)`
	`MODE AMODE(31),RMODE(ANY)`
	`NAME DFS3CDX0(R)`
	`/*`

Including the routine	No special steps required.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSMPL. **Note:** You must customize the sample exit routine before you can compile it.

Communicating with IMS

IMS communicates with this routine through the entry registers, a parameter list, and the exit registers. The exit routine must save all registers with the provided save area on entry. The exit routine must restore all registers before returning control to IMS.

Table 13. Contents of registers on entry

Register	Content
1	Address of the Version 6 standard exit parameter list.
13	Address of the exit save area. The exit routine must not change the first three words of the save area. This save area is not chained to any other save area.
14	Return address.
15	Entry point of this exit routine.

Register 1 contains the address of the Version 6 standard exit parameter list. The standard exit parameter list contains the field SXPLFSPL which is the address of the function-specific parameter list for the Catalog Definition exit. Some fields in the parameter list are directly equivalent to parameters in the DATABASE and CATALOG sections of the DFSDFxxx member of the IMS.PROCLIB data set.

For a description of the fields in the Version 6 standard exit parameter list, see IMS standard user exit parameter list (Exit Routines).

The function-specific parameter list is mapped by macro DFS3DXP and contains the following fields:

Table 14. Catalog Definition exit routine function-specific parameter list

Field	Offset	Length	Description	Equivalent DFSDF*xxx* parameter
DXPL_PVER	X'00'	4	Version number of the function-specific parameter list: **1** Version 1 **2** Version 2. Includes DXPL_FUNCV2 at offset X'60'. **3** Version 3. Includes DXPL_FUNCV3 at offset X’63’.	
DXPL_FUNC	X'04'	4	Function code: **1** Catalog enabled	CATALOG=YES (required)
DXPL_LEN	X'08'	4	Parameter list length	
DXPL_RGNTYPE	X'0C'	4	Region type: **1** Batch region	
DXPL_URCATL	X'10'	4	Unregistered catalog name list	UNREGCATLG (optional)
	X'14'	4	Reserved	
DXPL_RETNUM	X'18'	2	Number of catalog record copies to retain	RETENTION VERSIONS (optional)
DXPL_RETPD	X'1A'	2	Record retention period in days	RETENTION DAYS (optional)
DXPL_ALIAS	X'1C'	4	Alias name prefix	ALIAS (required)
	X'20'	8	Reserved	None
DXPL_DATC	X'28'	8	Data class	DATACLAS (optional)
DXPL_MGTC	X'30'	8	Management class	MGMTCLAS (optional)
DXPL_STGC	X'38'	8	Storage class	STORCLAS (optional)
DXPL_1PCT	X'40'	2	Primary data set space allocation percentage	SPACEALLOC PRIMARY (optional)
DXPL_2PCT	X'42'	2	Secondary data set space allocation percentage	SPACEALLOC SECONDARY (optional)
	X'44'	4	Reserved	
	X'48'	4	Reserved	
	X'4C'	4	Reserved	
	X'50'	4	Reserved	
	X'54'	4	Reserved	

Table 14. Catalog Definition exit routine function-specific parameter list (continued)

Field	Offset	Length	Description	Equivalent DFSDF*xxx* parameter
	X'58'	2	SMS volume count	SMSVOLCT (optional)
DXPL_VOL	X'5A'	6	Non-SMS primary or secondary index volume	IXVOLSER (required when the catalog data sets are not managed by SMS)
DXPL_FUNCV2	X'60'	1	**X'80'** IMS management of ACB is enabled	ACBMGMT=CATALOG
	X'61'	7	Reserved	
DXPL_FUNCV3	X’63’	1	**X'80'** Dynamic attach of IMS catalog PCBs is enabled.	CATPSBATTACH=YES
	X’63’	1	**X'40'** Dynamic attach of IMS catalog PCBs is disabled.	CATPSBATTACH=NO

Related reference

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

CCTL exit routines

The database resource adapter (DRA) can pass control to four coordinator controller (CCTL) exit routines, each of which may contain code provided entirely or in part by the CCTL.

If the CCTL passes an address (in the INIT request) of zero for a particular routine, the DRA uses a default exit routine.

Coordinator controller routine attributes

Coordinator controller (CCTL) routines have certain attributes and requirements.

All CCTL exit routines called by the database resource adapter (DRA) have control passed to them in 31-bit addressing mode and must return to the DRA in the same mode. Since much of the DRA has RMODE=31, registers 13 and 14 can point to locations above the 16 MB line. When the DRA calls the Control exit routine, the PAPL that it passes can also be above the line.

On entry to a CCTL exit routine, the PAPLTTOK and PAPLUSER fields are the same as they were when DFSPRRC0 first received the PAPL. (For more information on these fields, see *IMS Version 15.2 System Programming APIs*.) The CCTL uses the PAPLUSER field to pass information to the exit routines (for example, the address of the control blocks).

If you want the DRA to use the default exit routines supplied with IMS DB, pass a value of binary 0 as the address of the exit routine in the INIT request. For more information, see the topic “INIT request” in *IMS Version 15.2 System Programming APIs*.

To use the default Suspend exit routine and Resume exit routine, each DRA request must have the field PAPLTECB set with the address of a CCTL ECB to be used if the thread is waited or posted.

Suspend exit routine

The Suspend exit routine receives control whenever the database resource adapter (DRA) router routine needs to suspend a DRA request and allows the CCTL to use its own processing technique to suspend its thread.

The Suspend exit routine can start executing before or after the Resume exit routine starts executing, but the Suspend exit routine cannot finish executing before the Resume exit routine starts executing. When you design the Suspend and Resume exit routines, ensure that the Suspend exit routine can determine whether the Resume exit has started or completed execution. If the Suspend exit routine determines that the Resume exit routine has not started executing, the Suspend exit routine must not return to the caller. If the Suspend exit routine determines that the Resume exit routine has started or completed execution, the Suspend exit routine should return to the Suspend exit caller and consider the suspend request complete.

The Suspend exit routine executes in the CCTL's environment. The contents of the registers on entry are:

Register
Contents

1
Address of the PAPL

14
Return address

15
Entry point address

This routine can use a PAPL 16-word save area (PAPLSREG) to save the DRA's registers The DRA does not expect any output from this routine.

Resume exit routine

The Resume exit routine allows the CCTL to use its own processing technique to resume a database resource adapter (DRA) request suspended by the Suspend exit routine.

The Resume exit routine can start executing before or after the Suspend exit routine starts executing. When you design the Suspend and Resume exit routines, ensure that the Suspend exit routine can determine whether the Resume exit has started or completed execution. If the Suspend exit routine determines that the Resume exit routine has not started executing, the Suspend exit routine must not return to the caller. If the Suspend exit routine determines that the Resume exit routine has started or completed execution, the Suspend exit routine should return to the Suspend exit caller and consider the suspend request complete.

This routine receives control whenever a request has completed its process. The contents of the registers on entry are:

Register
Contents

1
Address of the PAPL

13
Address of an 18-word save area that this routine can use to save DRA registers

14
Return address

15
Entry point address

The DRA does not expect any output from this routine.

Control exit routine

The Control exit routine allows the database resource adapter (DRA) to notify the CCTL about events occurring within the DRA or IMS DB. It also allows the CCTL to notify the DRA how to respond to those events.

This routine receives control whenever the DRA must notify the CCTL of the following events:

- The DRA successfully identifies itself to IMS DB.
- The identify attempt to IMS DB fails.
- The CCTL's INIT request is canceled.
- The DRA fails.
- IMS DB fails.
- IMS DB terminates normally using the /CHECKPOINT FREEZE command.
- The DRA terminates due to a Control exit routine request.

The Control exit routine uses a PAPL that belongs to the DRA, never a CCTL PAPL that is a DRA request.

For all of these events (except the last one), the CCTL must tell the DRA what action to execute next. This is done using a return code that the CCTL places in the PAPLRETC field prior to passing the PAPL back to the DRA. The DRA then acts accordingly.

The contents of the registers on entry are:

Register
Contents

1
Address of the PAPL

13
Address of standard 18-word save area that the Control exit routine can use

14
Return address

15
Entry point address

A list of possible events about which the DRA notifies the CCTL follows. With each event, the contents of the PAPL are listed with possible actions for the CCTL to take.

Subsections:

The DRA successfully identifies itself to IMS DB

After the DRA successfully identifies to IMS DB, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Resync function code, PAPLRSYN

PAPLRSLT
Resync list address, list of recovery tokens of indoubt UORs. First 4 bytes in the list is the number of tokens in the list. Following this number are the actual tokens, each being 16 bytes.

PAPLUSER
User data (passed on the INIT request).

PAPLDBCT
IMS DB identifier.

PAPLMTCB
Minimum thread count specified in the startup table or INIT request.

PAPLJOBN
IMS DB jobname.

PAPLCRC
IMS DB command recognition character.

PAPLIDTK
IMS DB identify token (unique store clock value representing the time the CCTL identified with IMS DB).

PAPLDSID
IMS DB address space ID (ASID).

PAPLRSEN
DBRSE (IMS DB warm standby name, =DBRSENM, IMS DB execution parameter). See *IMS Version 15.2 System Definition* for more information.

PAPLRGTY
IMS region type. The possible region types are:

PAPLDBCX
DB/DC with XRF.

PAPLDBCO
DB/DC only.

PAPLDBCL
IMS DB

After the routine has completed analyzing the PAPL, it can insert the following return codes in the PAPLRETC field to notify the DRA of the next action to take:

Code Returned
Meaning

0
IMS DB environment OK.

4
Terminate the DRA (the Control exit routine is not called again during this DRA session).

The identify attempt to IMS DB fails

After the identify to IMS DB fails, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Failure function code

PAPLSFNC
Identify request failed subfunction code

PAPLUSER
User data (passed on the INIT request)

PAPLDBCT
IMS DB identifier

PAPLRETC
Code returned from subsystem interface or IMS DB

PAPLRCOD
Reason code. The possible reason codes are:

PAPLNTUP
Subsystem exists but is not up

PAPLNOSS
Subsystem does not exist

PAPLINT
IMS DB is in initialization process

PAPLRSTN
IMS DB waiting for restart command

PAPLRST
In restart process

PAPLBRST
DB/DC XRF backup in tracking mode

PAPLTKOV
Backup in takeover mode

After the routine analyzes the PAPL, it can insert the following data in the output fields in the PAPL to notify the DRA of the next action to take:

Field
Contents

PAPLDBCN
New IMS DB identifier

PAPLRETC
Code returned from the CCTL to the DRA. PAPLRETC is passed to the Control exit routine and must be reset.

Code Returned
Meaning

0
Issue a DFS0690A message and try to identify IMS DB again.

4
Proceed with DRA termination (the Control exit routine will not be called again).

8
Reidentify with new IMS DB identifier (in the PAPLDBCN field).

The CCTL's INIT request is canceled

After the DRA INIT request is canceled by a `cancel` response to the DRF690 message, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Failure function code

PAPLSFNC
Cancelled INIT request subfunction code

PAPLUSER
User data (from the INIT request).

PAPLDBCT
IMS DB identifier.

PAPLRETC
Code returned from IMS DB.

PAPLRCOD
Reason code. The possible reason codes are:

PAPLDBNZ
IMS DB rejected identify request.

PAPLOPC
Operator responded `cancel` to DFS690 message.

After the routine has completed analyzing the PAPL, it can insert the following return codes in the PAPLRETC field to tell the DRA what to do next:

Code Returned
Meaning

0
Wait for a DRA TERM request.

4
Proceed with DRA termination (the Control exit routine will not be called again).

PAPLRETC is passed to the Control exit routine and must be reset.

The DRA fails

When the DRA fails, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Failure function code

PAPLDRAF
DRA failure subfunction code.

PAPLUSER
User data.

PAPLDBCT
IMS DB identifier.

PAPLRCOD
Reason code

The reason codes possible are:

PAPLGMF
GETMAIN failed.

PAPLSSF
Subsystem interface failure.

PAPLDRAA
DRA abend.

PAPLESTF
Unable to establish DRA ESTAE.

The DRA expects no return code in PAPLRETC. The DRA fails and the Control exit routine is not called when the failure occurs while processing a TERM request. In this case, the PAPL return code of the returned TERM PAPL contains the failure code.

IMS DB fails

When IMS DB fails, the DRA first issues a U002 abend to all DRA thread TCBs. In some cases, the DRA itself can also get a U002 abend and call the Control exit routine as in the previous failure event. Otherwise, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Failure function code.

PAPLDBCF
IMS DB failure subfunction code.

PAPLUSER
User data.

PAPLDBCT
IMS DB identifier.

PAPLRETC
Code returned from IMS DB.

PAPLRCOD
Reason code. The reason code is:

PAPLABND
IMS DB abend.

The DRA expects no return code in PAPLRETC.

After the exit routine analyzes the PAPL, it can insert the following identifier and return codes in the output fields of the PAPL to notify the DRA of the next action to take:

Field
Contents

PAPLDBCN
New IMS DB identifier.

PAPLRETC
Code returned.

PAPLRETC is passed to the Control exit routine and must be reset.

Code Returned
Meaning

0
Wait for a DRA TERM request.

4
Wait for DRA termination.

8
Try to identify again with the new IMS DB identifier in the PAPLDBCN field.

IMS DB terminates normally using the /CHECKPOINT FREEZE command

After IMS DB terminates using a /CHECKPOINT FREEZE command, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Failure function code.

PAPLDBCC
IMS DB /CHE FREEZE subfunction code.

PAPLUSER
User data.

PAPLDBCT
IMS DB identifier.

After the exit routine analyzes the PAPL, it can insert the following identifier and return codes in the output fields of the PAPL to notify the DRA of the next action to take:

Field
Contents

PAPLDBCN
IMS DB identifier.

PAPLRETC
Code returned.

Code Returned
Meaning

0
Allow the DRA to shut itself down.

4
Terminate DRA immediately.

8
The current DRA threads are allowed to complete all current calls and are then terminated. The DRA then reidentifies with the new IMS DB identifier.

After the CCTL sets the return code equal to 0, the DRA follows the rules of the /CHECK FREEZE command (for example, it allows the current threads to complete their units of work). After the last thread completes, the DRA terminates. The invocation of the Control exit routine signals the completion of the DRA shutdown process.

The DRA terminates due to a Control exit routine request

After the DRA terminates due to a Control exit routine request, the contents of the PAPL passed to the CCTL are:

Field
Contents

PAPLFUNC
Failure function code.

PAPLDRAF
DRA failure subfunction code.

PAPLUSER
User data.

PAPLDBCT
IMS DB identifier.

PAPLRCOD
Reason code.

The possible reason codes are:

PAPLITCF
DRA terminated due to a Control exit routine request.

PAPLMXN2
Statistic #1 (see *IMS Version 15.2 System Programming APIs*)

PAPLMIN2
Statistic #2 (see *IMS Version 15.2 System Programming APIs*)

PAPLHIT2
Statistic #3 (see *IMS Version 15.2 System Programming APIs*)

PAPLTIM2
Statistic #4 (see *IMS Version 15.2 System Programming APIs*)

Since the DRA terminated, the CCTL does not pass any return codes to IMS DB.

Control is passed to this exit routine at the end of the DRA cleanup when the DRA termination is due to a previous Control exit routine request. For example, after being notified of a IMS DB failure or a /CHE FREEZE command, the Control exit routine terminates the DRA.

Status exit routine

The Status exit routine prevents a z/OS S0C4 abend from occurring when a CCTL thread attempts to access nonexistent storage.

The database resource adapter (DRA) passes control to the Status exit routine when a task control block (TCB) for a DRA thread in a scheduled state is collapsing.

The *scheduled state* is the time between the DRA's successful processing of a schedule request and the DRA's successful processing of one of the following thread function requests:

ABTTERM
Abort unit of work.

COMTERM
Commit unit of work.

TERMTHRD
Terminate thread.

Related Reading: Refer to the section on CCTL DRA function requests in *IMS Version 15.2 System Programming APIs* for a description of the thread functions.

The status exit is called to:

- Notify CCTL that the DRA thread is about to terminate for a reason other than a request from the CCTL.
- Allow CCTL to stop reference, by the CCTL thread, to storage that IMS DB acquired on behalf of the thread.
- Notify CCTL to free the storage that IMS DB acquired for the thread.

When a DRA thread successfully processes a schedule request, the address of the storage that IMS DB acquired in the CCTL's private storage is returned to the CCTL. The storage is acquired and initialized with the user's PCBLIST and PCBs. The CCTL thread uses the PCBLIST and PCBs to make DL/I requests and to receive the results of the requests. The storage is referred to as user private storage (UPSTOR).

Related Reading: See the topic on CCTL DRA function requests in *IMS Version 15.2 System Programming APIs* for PAPL fields returned to CCTL when the schedule request is completed.

The CCTL thread has access to UPSTOR for the duration of the thread's scheduled state. When the scheduled state terminates normally by a request from the CCTL, IMS DB manages UPSTOR storage.

Reference to UPSTOR by the CCTL thread after the normal end of a scheduled state can result in a z/OS S0C4 abend if IMS DB has freed the storage. If IMS DB allocated the same storage to another thread, reference to UPSTOR can overlay the second thread's data.

When the thread terminates abnormally during the scheduled state, the Status exit routine notifies the CCTL. The CCTL is responsible for freeing UPSTOR. The responsibility for freeing UPSTOR is assigned to the CCTL to ensure that UPSTOR is freed at the proper time.

The UPSTOR area is acquired using the GETMAIN macro by DRA thread TCBs out of subpool 0 (subpool 132 if the CCTL application is running with the public key option set).

The default Status exit routine provided by the DRA frees UPSTOR. If the CCTL chooses the default exit routine, it can incur a program check abend trying to access that storage because the CCTL might execute after the DRA has freed the storage.

The contents of the registers on entry are:

Register
Contents

1
Address of the PAPL.

13
Address of standard 18-word save area that the Status exit routine can use.

14
Return address.

15
Entry point address.

If DRA thread termination occurs during processing of a CCTL request, the CCTL's PAPL is passed to the Status exit routine. Otherwise, the DRA builds a PAPL.

The contents of the PAPL that are significant for the call are:

Field
Contents

PAPLUSR3
The value CCTL passed in PAPLUSR3 on the INIT request.

PAPLTOKT
The thread token set up by the CCTL. This is the token which the CCTL passed, in PAPLTTOK, on the SCHED request.

PAPLUPSA
Address of UPSTOR.

PAPLUPSL
Length of UPSTOR.

The DRA expects no return code in the field PAPLRETC.

Data Capture exit routine

You can write a Data Capture exit routine that receives control whenever a segment, for which the exit routine is defined, is updated. Your exit routine processes the data after the DL/I call completes but before control is returned to the application program.

This topic contains Product-sensitive Programming Interface information.

When an application program updates an IMS database with a DL/I insert, replace, or delete call, the original and updated data, as applicable, are passed and made available to a Data Capture exit routine. The DL/I call is considered complete and the PCB status is set when the exit routine is called. The following figure shows how control passes among the application, the full-function or DEDB database, and the exit routine.

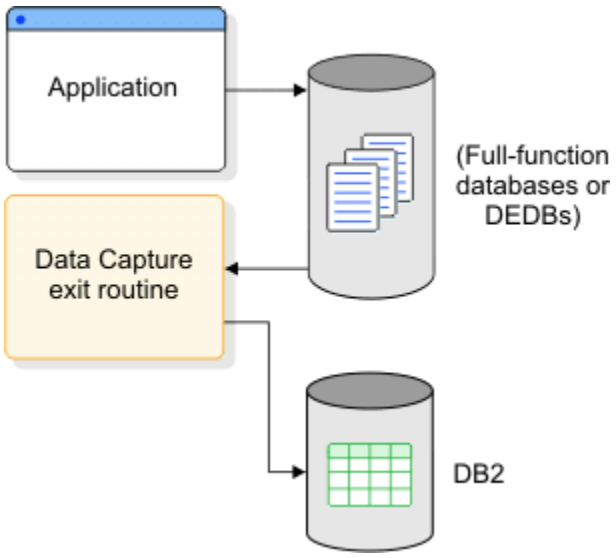


Figure 1. Calling order with data capture

You might want to capture changed data so that you can replicate that data to a Db2 for z/OS database as shown in the previous figure.

As an alternative to capturing data synchronously, you can also propagate captured data asynchronously by using either of the following methods:

- Use the logging option on the EXIT= parameter of DBDGEN.
- Use IMS DataPropagator and specify that the data is to be propagated asynchronously.

The following table describes data capture support for IMS environments for both full-function and DEDB databases.

Table 15. Data capture support for IMS environments

	CICS® DB/CTL	CICS Batch	IMS Batch	IMS IFP	IMS BMP	IMS MPP
Data Capture Exit EXIT=*exit_name*	No	Yes[1]	Yes	Yes	Yes	Yes
Asynchronous Data Capture EXIT= *, LOG	Yes	Yes[1]	Yes	Yes	Yes	Yes

Note: [1]BATCH is a pure IMS batch environment that is available with CICS DB/CTL (no CICS code executing).

Subsections:

- "About this routine" on page 62
- "Communicating with IMS" on page 64
- "Extended Program Communication Block (XPCB)" on page 66
- "Extended Segment Data Block (XSDB)" on page 68
- "Writing the routine in supported languages" on page 69
- "Storage requirements for Data Capture" on page 70
- "Storage failure" on page 71
- "Data security and integrity" on page 71

About this routine

The main purpose of capturing updated data and making it available to an exit routine is to propagate the IMS data to the relational environment of Db2 for z/OS. You can write your own exit routine, use a separate product, use IBM IMS DataPropagator for z/OS, or write a IMS DataPropagator-supported exit routine. If you write your own exit routine, you can code it to perform tasks other than data propagation. The sample Data Capture exit routine provided at the end of this topic only propagates data.

Restriction: This exit routine cannot be used with CICS, because it conflicts with CICS architecture. (Asynchronous Data Capture does work with DBCTL.) Even though the exit routine works with captured IMS data, CICS cannot use it.

Attributes of the routine

Regardless of its function, you must write the routine in assembly language, C language, COBOL, or PL/I. Routines written in high-level languages running under Language Environment for z/OS are not supported. Sample exit routines are provided in COBOL and PL/I.

Running Data Capture exit routines under Language Environment for z/OS might result in performance problems unless the dependent region that is running the application that causes the Data Capture exit routine to execute is pre-initialized in the Language Environment for z/OS. This can be done with the preinitialization list. Otherwise, every execution of the application in a dependent region causes the Language Environment for z/OS to be initialized each time the application is invoked and stopped each time the application terminates.

Binding the routine

If you bind the exit routine as either RENT or REUSE, it remains in storage until the region terminates as if the exit routine was preloaded. However, non-REUSE exit routines must be loaded each time, because they are deleted from storage after each call.

Loading the routine

IMS loads the exit routine the first time IMS calls it; preloading the exit routine is not necessary. However, runtime library routines used by high-level languages should be preloaded. After abnormal termination in an IMS Fast Path region (IFP) or in a message processing region (MPP), the exit routine is deleted and must be reloaded. The exit routine must be reloaded when:

- A pseudo or standard abend of the application that is running in the region occurs (regardless of whether the region itself abends along with the application).
- The data capture routine gets an XPCB return code of 16.

Specifying data options

In addition to the necessary control information, you can have the following data passed to your exit routine. The data is chained together using pointers.

Physical concatenated key
The fully concatenated key of each segment in the physical hierarchy, including the updated segment. For logical relationships and secondary indexes, this key differs from the key in the PCB feedback area.

Physical segment data
The physical segment updated by the application program, without any PSB field sensitivity.

Data before a replace
The data as it looked before it was updated. Your exit routine must determine what fields the application program changed.

Path data
The physical path data from the root segment to the parent of the updated segment.

Cascade delete data
The data deleted by IMS when an application program deleted a segment that is higher in the hierarchy.

The data is in the same format that was returned to the application program, excluding PSB field sensitivity. For logical children, the segment data follows the logical parent concatenated key. For segments with compression/edit exit routines defined for them, the data is in its expanded or encoded form. For variable-length segments, the first two bytes contain the length ('LL') for the segment.

Additional guidelines

The Data Capture exit routine is called whenever a segment is updated that has the exit routine defined, regardless of the execution environment. The exit routine uses the INQY ENVIRON call to identify the execution environment (batch or online) and determine what functions are available.

The exit routine can issue any DL/I calls allowed by the PSB using the AIB Interface (AIBTDLI). However, any updates that the exit routine makes are not captured and do not call an exit routine.

The Data Capture exit routine is treated as an extension of the application program; IMS attributes SQL or DL/I calls made by the exit routine to the application program. The exit routine and the application run under the same unit of work. SQL and DL/I updates made by the exit routine are committed or aborted along with the application program at sync-point time with the same integrity as the application. The exit routine must follow the same rules as the application program whether the routine makes IMS or Db2 for z/OS requests.

For data propagation, all DL/I updates must be passed to the exit routine to determine whether to propagate the change to Db2 for z/OS or not. Both the IMS data and Db2 for z/OS data must be available and on the same z/OS system for either update to occur.

The Data Capture exit routine is called based on specification in the DBD rather than in the PSB. Unless otherwise defined, the exit routine is always called: when implemented for any segment or database, all activity in that segment or database causes IMS to call the exit routine, regardless of which PSB is active. In this scenario, any performance impact that the exit routine causes occurs across the entire system. However, if you do not require the Data Capture exit routine to be called for the activity of a specific CCTL or ODBM address space, you can specify the job name of the CCTL or ODBM address space on the **SUPPDCAPNAME=** parameter in the DFSDF*xxx* member. If a job name of a CCTL or ODBM address space is specified on the **SUPPDCAPNAME=** parameter, database updates that are invoked by the job are not captured, even if a Data Capture exit routine is specified on the DBD.

Defining the routine for segments

The Data Capture exit routine is specified for a particular segment during DBDGEN. Failure to locate the exit routine during processing results in an application program abend.

DBDGEN supports the parameter, EXIT=, on the DBD and SEGM statements. If specified on the DBD statement, the parameter applies to all segments within the physical database structure. If specified on the SEGM statement, you can override the specification on the DBD, or can limit the parameter so that only selected segments are propagated when updated. As a SEGM parameter, EXIT= does not apply to other segments; physical children do not inherit the parameters of any of their parents.

You can specify multiple exit routine names, each with different data options, on a single DBD or SEGM statement.

Multiple exit routines

A single DL/I call might call your exit routine more than once or it might call more than one exit routine. Multiple exit routines are called when there are:

- Multiple exit routines per segment
- Path calls
- Cascade deletes

Multiple exit routines are called in succession before returning to the application program. The sequence depends on the reason multiple exit routines are called:

- Multiple exit routines are defined.

When multiple exit routines are defined for a single physical segment, the routines are called based on DBDGEN definition order. The first exit routine listed in the DBD or SEGM statement is called, followed by each subsequent exit routine defined for that segment.

- Multiple segments are updated.

  When multiple physical segments are updated in a single call, the routines are called in hierarchical order. IMS calls the exit routines for the segments in the same order that the segments were physically updated:

  – Top-down for path inserts and path replaces:

    Parents must be inserted before dependents. The exit routine for the parent segment must be called before the dependent segment's exit routine.

  – Bottom-up for cascade deletes:

    The dependent segment's exit routine is called before the parent's exit routine. The root segment's exit routine is called last. If the dependent segment has several exit routines defined for it, they are **all** called at this time. Calling the exit routines in bottom-up order allows propagation to Db2 for z/OS without requiring referential integrity.

    For each segment type, multiple segment occurrences might be deleted as part of the cascade delete. Each exit routine is called once for each segment occurrence that is deleted. The order the exit is called is the same order in which DL/I deleted the segments.

Using IMS callable services with this routine

This exit is not eligible to use IMS callable services.

Communicating with IMS

Each segment that is passed in a dependent region and has the Data Capture exit routine defined for it has two control blocks available for its use. Both the Extended Program Communication Block (XPCB) and the Extended Segment Data Block (XSDB) reside in private storage and have key 8. They are passed to the exit routine according to the AMODE of the exit: above the 16 MB line for AMODE 31, and below the 16 MB line for AMODE 24.

The order in which the control blocks receive control depends on the type of data updated and passed to the Data Capture exit routine. The following figure shows how control flows between the XPCB and the XSDB.

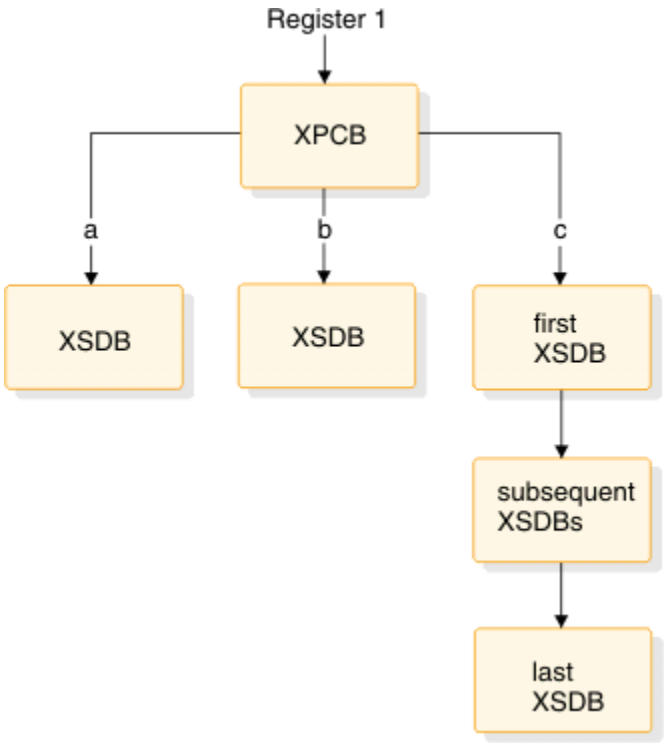


Figure 2. Control block flow with data capture

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the XPCB address
13	Address of save area
14	Return address to IMS
15	Entry point of exit routine

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers. Return and reason codes are placed in the XPCB.

Return and reason codes

The XPCB contains fields for the exit routine to communicate its status to IMS. These fields are initialized to binary zeros. The return code set by the exit routine defines the type of condition encountered; the higher the number, the more severe the error. You can also assign a reason code to return codes of 8 or greater. The reason code is for your use; IMS uses only the return code.

The following table outlines the return and reason codes that the exit routine returns and places in the XPCB. If the return code placed in the XPCB is invalid, an abend occurs and an invalid return code indicator is set.

Table 16. XPCB return codes

Return code	Description	Action	DFS3314 message
0	Good return.	Normal completion of exit routine.	No

Table 16. XPCB return codes (continued)

Return code	Description	Action	DFS3314 message
4	Indicates the exit routine wants to ignore the DL/I call.	Exit routine is not called for any additional segments for this DL/I call.	No
8	Exit routine encountered an error during the DL/I call and wants to return to the application.	DL/I call is terminated without calling any other exit routines and control is returned to application program.	Yes
12	This copy of the exit routine is not to be called again. (Used with a "dummy" exit routine.)	Exit routine is deleted from storage.	Yes
16	Abend the exit routine and the application program.	Application program is abended with a U3314.	Yes
20	Do not make further calls to this routine, or any other Data Capture routines, for this region.	Terminate data capture for this region.	Yes

After an abend in an IFP or MPP region with return code 12 or 20, the interface control blocks are reinitialized and the exit work area is reset. The exit routine can then be called again.

Extended Program Communication Block (XPCB)

The XPCB identifies the segment and call functions, provides the address of a work area, and contains additional information that is passed to the exit routine. Every XPCB identifies the physical function performed by DL/I (insert, replace, or delete) and points to the updated data that is passed to the exit routine. The following two tables describe the contents of the XPCB.

For reentrant exit routines, the address of a 256-byte work area is passed in the XPCB. The exit routine can use the work area to save information. One work area exists for each exit routine, and it is initialized to binary zeros the first time the exit routine is given control.

Table 17. XPCB by offset

Offset	Field name	Offset	Field name	Offset	Field name
0	Eye catcher	4	Version	6	Release
8	User_Exit_Name	16	Exit_Return_Code	18	Exit_Reason_Code
20	Database_Name	28	DBD_Version_Ptr	32	Segment_Name
40	Call_Function	44	Physical_Function	48	reserved
52	DB_PCB_Ptr	56	DB_PCB_Name	64	INQY_Output_Ptr
68	IO_PCB_Ptr	72	Environment_Flags	73	reserved
74	Conc_Key_Length	76	Conc_Key_Ptr	80	Data_XSDB_Ptr
84	Before_XSDB_Ptr	88	Path_XSDB_Ptr	92	Set_Rols_Token
96	Next_Twin_Ptr	100	Cmd_Codes_Ptr	104	Exit_Work_Ptr
108	Null_Ptr	112	reserved	116	Call_Timestamp

Table 18. XPCB alphabetically

Field name	Offset	Data type	Length	Field description
Before_XSDB_Ptr	84	Pointer	4	Address of XSDB for data before it was replaced. Zero if not a physical replace or if data not captured.
Call_Function	40	Character	4	Call used by application to update segment: ISRT, DLET, REPL, FLD (field), or CASC (cascade).
Call_Timestamp	116	Character	8	Time stamp of completion of DL/I call. Obtained from Store Clock instruction.
Cmd_Codes_Ptr	100	Pointer	4	Address of command codes. This field points to a data area that has the same format as the COMMAND_CODES in the CAPD block format.
Conc_Key_Length	74	Fixed	2	Length of the segment concatenated key for physical path. Zero if data not captured. Key is optional.
Conc_Key_Ptr	76	Pointer	4	Address of the segment concatenated key for physical path. Zero if data not captured. Key is optional.
Database_Name	20	Character	8	Name of physical database that contains the updated segment.
Data_XSDB_Ptr	80	Pointer	4	Address of XSDB for segment data. Zero if data not captured.
DBD_Version_Ptr	28	Pointer	4	Address of variable length character string to identify the DBD used for update. First 2 bytes contain length of string, followed by string itself. String is from DBD VERSION= parameter if it was used for DBDGEN. Otherwise, string is date/time of DBDGEN.
DB_PCB_Ptr	52	Pointer	4	Address of database PCB used for DL/I call.
DB_PCB_Name	56	Character	8	The 8-byte name of database PCB used for DL/I call. Null if name not assigned during PSBGEN with the label or PCBNAME= parameter.
Environment_Flags	72	Flag byte	1	Flag bits describing execution environment.
Exit_Return_Code	16	Fixed	2	Return code from exit routine.
Exit_Reason_Code	18	Fixed	2	Reason code from exit routine.
Exit_Work_Ptr	104	Pointer	4	Address of 256-byte work area.
Eye catcher	0	Character	4	'XPCB'
INQY_Output_Ptr	64	Pointer	4	Address of output of an INQY ENVIRON call.
IO_PCB_Ptr	68	Pointer	4	Address of I/O PCB.
Next_Twin_Ptr	96	Pointer	4	Address of XSDB for the data of the twin that follows the segment being inserted. Zero if not a twin or if no other twins exist for the non-unique segment.

Table 18. XPCB alphabetically (continued)

Field name	Offset	Data type	Length	Field description
Null_Ptr	108	Pointer	4	Zero address for use as null address for languages that do not recognize a zero address as null (such as PL/I).
Path_XSDB_Ptr	88	Pointer	4	Address of XSDB for physical root when path data option requested. XSDBs for path data are chained together, in descending hierarchical order, from physical root to parent of updated segment. Last XSDB has a zero pointer.
Physical_Function	44	Character	4	Physical call function performed: ISRT, DLET, or REPL.
Release	6	Character	2	XPCB release indicator. Along with version, identifies the level of the control block. The current release is R3.
Segment_Name	32	Character	8	Physical segment name of segment updated.
Sets_Rols_Token	92	Hexadecimal data	4	Token that is used to identify the processing scope between the SETS and ROLS calls.
User_Exit_Name	8	Character	8	Entry point name of exit routine.
Version	4	Character	2	XPCB version indicator. Along with release, identifies the level of the control block. The current version is V1.

Extended Segment Data Block (XSDB)

The XPCB points to the first XSDB. For path data, subsequent XSDBs are chained together. The XSDB points to the updated data that is passed to the exit routine. It contains additional information that is also passed. The following two tables describe the contents of the XSDB.

Table 19. XSDB by offset

Offset	Field name	Offset	Field name	Offset	Field name
0	Eye catcher	4	Version	6	Release
8	Next_Ptr	12	Database_Name	20	Segment_Name
28	Physical_Path	29	CMD_CODE_R	30	reserved
32	Segment_Level	34	Key_Length	36	Key_Ptr
40	LP_Key_Length	42	Segment_Length	44	Segment_Ptr
48	reserved				

Table 20. XSDB alphabetically

Field name	Offset	Data type	Length	Field description
CMD_CODE_R	29	Flag byte	1	Subset pointer command codes R1 through R8. Each bit represents whether or not the corresponding command code number was specified on the SSA.
Database_Name	12	Character	8	Name of physical database that contains the updated segment.

Table 20. XSDB alphabetically (continued)

Field name	Offset	Data type	Length	Field description
Eye catcher	0	Character	4	'XSDB'
Key_Length	34	Fixed	2	Length of key for segment. Zero if segment not keyed.
Key_Ptr	36	Pointer	4	Address of key for segment. Zero if segment not keyed.
LP_Key_Length	40	Fixed	2	Length of the concatenated key of a logical parent segment included in segment data for logical children.
Next_Ptr	8	Pointer	4	Address of next XSDB in chain for path data. Zero for last XSDB in chain.
Physical_Path	28	Character	1	Access by physical path (Y/N)
Release	6	Character	2	XSDB release indicator. Along with version, identifies level of control block. The current release is R2.
Segment_Ptr	44	Pointer	4	Address of physical segment data.
Segment_Length	42	Fixed	2	Length of physical segment data.
Segment_Level	32	Fixed	2	Level of segment in physical database.
Segment_Name	20	Character	8	Physical segment name for segment data passed in this block. Different from segment name in XPCB for path data.
Version	4	Character	2	XSDB version indicator. Along with release, identifies level of control block. The current version is V1.

Writing the routine in supported languages

Although the Data Capture exit routine can be written in assembler language, C, COBOL, or PL/I, you must follow certain guidelines depending on which language you use.

Assembler

The exit routine is entered in primary mode, but the access registers can be nonzero.

C

C does not support variable-length character strings using integer lengths, such as those passed in the XPCB and XSDB. Key and segment data passed to the exit routine is terminated by "null" (binary zero) values. Any null value in the data itself might result in an invalid string length.

The following declarations and statements are used to locate the XPCB. Declare XPCB_TYPE_PTR as a pointer to the XPCB structure.

```
XPCB_TYPE_PTR *TPTR;
TPTR = (XPCB_TYPE_PTR *) __sysplist;
XPCB = *TPTR;
```

The exit routine must be defined as a main program with the PLIST(IMS) and ENV(IMS) options specified. Use the following format to specify these options:

```
#pragma runopt(env(IMS), plist(IMS))
```

COBOL

The exit routine operates under a separate run unit from the application program. The method used to establish the run unit depends on the compiler or on the RES/NORES compiler option. For all COBOL programs compiled with newer compilers, and older COBOL programs compiled with resident (RES), the

exit routine is given control by LINK. For older COBOL programs compiled with nonresident (NORES), it is given control directly.

Recommendation: Use a compiler with RES and code the exit routine as reentrant (RENT) and AMODE 31. With older compilers and NORES, the routine must be AMODE 24 and it must not be reentrant.

Attention: You can use GOBACK to terminate the exit routine run unit and return to the application program, but do not use STOP RUN and EXIT PROGRAM because they are not supported and might cause unpredictable results or abends.

The procedure division is:

```
exitname USING XPCB
```

PL/I

The exit routine must be compiled as a main program. The entry point can be PLICALLA, so that the exit routine can use the assembler interface or use PL/I compile-time option SYSTEM(IMS)

The procedure statement is:

```
exitname:  PROCEDURE(XPCB_PTR) OPTIONS (MAIN);
```

Storage requirements for Data Capture

As your application program issues a DL/I call to update the database, the updates are stored as required for use by the Data Capture exit routine or the Asynchronous Data Capture. Because the amount of storage required can be significant for update functions like a cascade delete, a data space is acquired for each dependent region that uses the exit routine. The attributes of the data space vary for online and batch-dependent regions, as illustrated in the following table.

Table 21. Data space characteristics (Data Capture exit routine and Asynchronous Data Capture)

Attribute	Online Dependent Region	Batch Dependent Region
Number of data spaces	1 per dependent region	1
Data space name	SYSDFS01	@SYSDFS1
Storage key	Key 7, not fetch protected to allow access from dependent region in key 8	Key 8
Storage size	By region controller	By region controller. Default size used if space requested violates total size of key 8 data spaces.
Storage obtained	During region initialization	During region initialization if exit routines are defined
Storage owned	By region controller TCB	By batch TCB
Added to access list	Dependent region address space, for access by program controller TCB in message regions. Control regions SAS address space for access by DL/I in an IMS DB/DC system when data capture is required. DEDB capture runs under program controller TCB.	Batch TCB
Deleted from access list	Dependent region always accessed. Deleted from control region SAS access list during thread termination if added to access list by data capture.	Not deleted
Data space cleared	During normal thread termination for message regions if data space storage was referenced.	Not cleared

Table 21. Data space characteristics (Data Capture exit routine and Asynchronous Data Capture) (continued)

Attribute	Online Dependent Region	Batch Dependent Region
Data space deleted	At region termination.	At z/OS job termination

You can control the use of data spaces with the SMF IEFUSI Step Initiation exit routine for key 8 batch regions. This exit routine determines the number and size of the data space available for key 8. If you have batch application programs that call the Data Capture exit routine, the data space specified for key 8 must be large enough to accommodate the data space requirements of data capture.

Storage failure

The two types of storage failure for data capture are:

- Data space not obtained. This type of error occurs in batch regions when a data space is not specified for each region. Online dependent regions can always obtain data space.
- Insufficient storage in the data space. In online dependent regions, storage space is specified by the region controller. Some database functions, such as cascade delete, require more than the space allocated for successful completion. Batch dependent regions can be limited in data space size. You must specify a data space large enough for data capture to complete successfully.

Either type of storage failure terminates the region with a U814 abend.

Data security and integrity

The Data Capture exit routine is an extension of the application program with the same capabilities as the application program; the exit routine and the application have equal authorization and limitations. IMS and Db2 for z/OS resources that the exit routine uses must be authorized in the IMS PSB or DB2® PLAN for the application program. This behavior ensures that the application program can access any IMS or Db2 for z/OS data that is available to the exit routine.

The data and the exit routine operate in unprotected, key-8 storage. The exit routine is able to modify data or control blocks that can affect the successful operation of the application program. The data passed to the exit routine is the physical segment data. With PSB field sensitivity, this data might include data that is unavailable to the application.

Related concepts

Data Capture exit routines (Database Administration)

Asynchronous data propagation (System Programming APIs)

z/OS: Dynamic Exits Facility

IMS DataPropagator Introduction

Related reference

INQY call (Application Programming APIs)

DATABASE section of the DFSDFxxx member (System Definition)

Examples of the DBDGEN utility (System Utilities)

Related information

0814 (Messages and Codes)

Sample Data Capture exit routine

A Data Capture exit routine can receive control whenever a segment, for which the exit routine is defined, is updated.

This topic provides examples of the Data Capture exit routine in COBOL and PL/I. The exit routine can also be written in assembler or C.

Subsections:

- “COBOL” on page 72

- “PL/I” on page 74

COBOL

The following example is the Data Capture exit routine in COBOL.

```
       IDENTIFICATION DIVISION.
         PROGRAM-ID. DLICDCE.
      *---------------------------------------------------------------*
      *REMARKS.                                                       *
      *---------------------------------------------------------------*
      *    DESCRIPTIVE NAME : HOSPITAL DATA BASE SEGMENT EXIT         *
      *---------------------------------------------------------------*
      *    THIS IS A SAMPLE IMS EXIT. THIS WILL BE CALLED BY IMS.     *
      *    THIS PROGRAM PROPAGATES DATA FROM IMS TO DB2 SYNCHRONOUSLY.*
      *    THE NAME OF THIS PROGRAM LOAD MODULE IS SPECIFIED          *
      *    ON SEGM MACRO DURING DBDGEN FOR THE HOSPITAL DATA BASE.    *
      *                                                               *
      *    THE DATA OPTIONS SELECTED FOR THIS EXIT :                  *
      *    EXIT=(KEY,DATA,NOPATH,CASCADE)                             *
      *---------------------------------------------------------------*
      *    INPUT FOR THIS PROGRAM : XPCB, XSDB.                       *
      *                                                               *
      *    OUTPUT:  DISPLAY A MESSAGE WHEN THE IMS UPDATE IS NOT      *
      *             ISRT, REPL, DELE, CASC. DISPLAY 'SQLERRM' WHEN    *
      *             SQLERROR OCCURS.                                  *
      *                                                               *
      *    UPDATES: UPDATES DB2 ILLNESS TABLE                         *
      *---------------------------------------------------------------*
      *    LOGIC:   THIS PROGRAM IS CALLED BY IMS AFTER THE IMS UPDATE*
      *             TO ILLNESS SEGMENT AND BEFORE IMS RETURNS TO THE  *
      *             IMS APPLICATION PROGRAM.                          *
      *                                                               *
      *             XPCB IS RECEIVED AS INPUT TO THIS PROGRAM.        *
      *             IF THERE IS NO ADDRESS OF XSDB IN XPCB THIS       *
      *             PROGRAM WILL RETURNS TO IMS OTHERWISE -           *
      *                                                               *
      *    LOGIC:   THIS PROGRAM IS CALLED BY IMS AFTER THE IMS UPDATE*
      *             WE GET THE ADDRESS OF XSDB FROM XPCB, FROM XSDB   *
      *             WE GET THE ADDRESS OF ILLNESS SEGMENT CONCATENATED*
      *             KEY, AND ADDRESS OF THE PHYSICAL SEGMENT DATA     *
      *                                                               *
      *             UPDATE THE DB2 ILLNESS TABLE WITH THE UPDATED IMS *
      *             SEGMENT DATA.                                     *
      * --------------------------------------------------------------*
       INSTALLATION.  IBM - SANTA TERESA LABORATORY.
       DATE-WRITTEN.  JANUARY 1990.
       ENVIRONMENT DIVISION.

       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-3090.
       OBJECT-COMPUTER. IBM-3090.
       DATA DIVISION.

       WORKING-STORAGE SECTION.
            EXEC SQL
              INCLUDE SQLCA
            END-EXEC.      *--- DB2 ILLNESS TABLE DECLARATION

            EXEC SQL
              DECLARE SYSADM.ILLNESS TABLE
                (ILLDATE VARCHAR (6)    NOT NULL,
                 PATNO   VARCHAR (5)    NOT NULL,
                 ILLNAME VARCHAR (10)   NOT NULL)
            END-EXEC.
      *---

       01  W-POINTER                      POINTER.
       01  W-POINTER-R REDEFINES W-POINTER PIC 9(8) COMP.
       LINKAGE SECTION.
      *--- EXIT SEGMENT CONTROL BLOCK

       01  XPCB.
           05  EYECATCHER                   PIC X(04).
           05  VERSION                      PIC X(02).
           05  RELEASE-ID                   PIC X(02).
           05  EXIT-NAME                    PIC X(08).
           05  EXIT-RETURN-CODE             PIC 9(04) COMP.
           05  EXIT-REASON-CODE             PIC 9(04) COMP.
```

```
           05  DATABASE-NAME                  PIC X(08).
           05  DBD-VERSION-PTR                POINTER.
           05  SEGMENT-NAME                   PIC X(08).
           05  CALL-FUNCTION                  PIC X(04).
           05  PHYSICAL-FUNCTION              PIC X(04).
           05  FILLER                         PIC 9(08) COMP.
           05  DB-PCB-PTR                     POINTER.
           05  DB-PCB-NAME                    PIC X(08).
           05  INQY-OUTPUT-PTR                POINTER.
           05  IO-PCB-PTR                     POINTER.
           05  ENVIRONMENT-FLAGS              PIC X.
             88  IMS-ENH-SUPPORT            VALUE X'80'.
       * RRS SUPPORT IS AVAILABLE IN SYSTEM
             88  IMS-RRS-ENABLED            VALUE X'40'.
       * RRS=Y WAS SPECIFIED
             88  CALL_AT_COMMIT             VALUE X'20'.
       * SET BY EXIT - CALL DURING COMMIT
             88  XPCB_LOGX_FORMAT           VALUE X'10'.
       * REDUCED 9904 FORMAT
             88  XPCB_EXIT_WAS_CALLED       VALUE X'08'.
       * INTERNAL FLAG USED BY IMS
             88  XPCB_DPROP_EXIT            VALUE X'04'.
       * SET BY DPROP EXIT ROUTINE
           05  FILLER                           PIC X.
       * RESERVED
           05  CONC-KEY-LENGTH                  PIC 9(04) COMP.
           05  CONC-KEY-PTR                     POINTER.
           05  DATA-XSDB-PTR                    POINTER.
           05  BEFORE-XSDB-PTR                  POINTER.
           05  PATH-XSDB-PTR                    POINTER.
           05  FILLER                           POINTER.
           05  FILLER                           POINTER.
           05  FILLER                           POINTER.
           05  EXIT-WORK-PTR                    POINTER.
           05  NULL-PTR                         POINTER.
           05  FILLER                           POINTER.
           05  TIMESTAMP                        PIC X(08).
      *--- EXIT SEGMENT DATA BLOCK

        01  DATA-XSDB.
           05  EYECATCHER                     PIC X(4).
           05  VERSION                        PIC X(2).
           05  RELEASE-ID                     PIC X(2).
           05  NEXT-PTR                       POINTER.
           05  DATABASE-NAME                  PIC X(8).
           05  SEGMENT-NAME                   PIC X(8).
           05  FILLER                         PIC X(4).
           05  SEGMENT-LEVEL                  PIC 9(4) COMP.
           05  KEY-LENGTH                     PIC 9(4) COMP.
           05  KEY-PTR                        POINTER.
           05  FILLER                         PIC 9(4) COMP.
           05  SEGMENT-LENGTH                 PIC 9(4) COMP.
           05  SEGMENT-DATA-PTR               POINTER.
           05  FILLER                         POINTER.
           05  FILLER                         POINTER.
      *--- ILLNESS SEGMENT DATA

       01  LS-SEGMENT.

           03  LS-ILLDATE                     PIC X(6).
           03  LS-ILLNAME                     PIC X(10).
      *--- ILLNESS SEGMENT CONCATENATED KEY

       01  XPCB-CONCKEY.

           02  LS-PATNO                       PIC X(5).
           02  LS-ILLDT                       PIC X(6).
       PROCEDURE DIVISION USING XPCB.
           SET W-POINTER TO DATA-XSDB-PTR.
      *--- LENGTH ZERO IF NOT CAPTURED

      *    IF W-POINTER-R EQUAL ZEROES GOBACK
      *       GOBACK
      *    END-IF
      *----

           SET  ADDRESS OF DATA-XSDB TO DATA-XSDB-PTR.
           SET  ADDRESS OF XPCB-CONCKEY TO CONC-KEY-PTR.
           SET  ADDRESS OF LS-SEGMENT TO SEGMENT-DATA-PTR.
      *----
             EXEC SQL
               WHENEVER SQLWARNING CONTINUE
```

```
            END-EXEC
            EXEC SQL
              WHENEVER SQLERROR GO TO BADSQL
            END-EXEC
            EXEC SQL
              WHENEVER NOT FOUND GO TO BADSQL
            END-EXEC
      *----

              IF PHYSICAL-FUNCTION OF XPCB = "ISRT"

                 EXEC SQL
                   INSERT INTO SYSADM.ILLNESS
                   VALUES (::LS-ILLDATE,::LS-PATNO,
                   ::LS-ILLNAME)
                  END-EXEC              ELSE
                 IF PHYSICAL-FUNCTION OF XPCB = "CASC" OR
                    PHYSICAL-FUNCTION OF XPCB = "DLET"
                    EXEC SQL
                      DELETE FROM SYSADM.ILLNESS
                      WHERE (PATNO = ::LS-PATNO AND
                             ILLDATE = ::LS-ILLDATE)
                    END-EXEC
                ELSE

                  IF PHYSICAL-FUNCTION OF XPCB = "REPL"

                     EXEC SQL
                       UPDATE SYSADM.ILLNESS
                       SET ILLNAME = ::LS-ILLNAME
                       WHERE (ILLDATE = ::LS-ILLDATE AND
                              PATNO = ::LS-PATNO)
                     END-EXEC
                   ELSE

                     DISPLAY "FUNCTION WASNT ISRT, REPL, DLET, OR CASC"
                     DISPLAY "--- NO SQL ACTION WAS TAKEN"
                     DISPLAY "PHYS FUNCTION IS "
                     DISPLAY PHYSICAL-FUNCTION OF XPCB
                   END-IF
                 END-IF
              END-IF.
              DISPLAY "SQLCODE " SQLCODE.
              GOBACK.
       BADSQL.

           DISPLAY "SQLERRM".
           MOVE 8        TO EXIT-RETURN-CODE OF XPCB.
           MOVE SQLCODE  TO EXIT-REASON-CODE OF XPCB.
           GOBACK.
```

PL/I

The following example is the Data Capture exit routine in PL/I.

```
DLI2DB2: PROCEDURE(XPCB_PTR) OPTIONS(MAIN);
/*
 *-------------------------------------------------------------------*
 *REMARKS.                                                           *
 *-------------------------------------------------------------------*
 *    DESCRIPTIVE NAME : HOSPITAL DATA BASE SEGMENT EXIT             *
 *-------------------------------------------------------------------*
 *    THIS IS A SAMPLE IMS EXIT THAT WILL BE CALLED BY IMS.          *
 *    THIS PROGRAM PROPAGATES DATA FROM IMS TO DB2 SYNCHRONOUSLY.*
 *    THE NAME OF THIS PROGRAM LOAD MODULE IS SPECIFIED              *
 *    ON SEGM MACRO DURING DBDGEN FOR THE HOSPITAL DATA BASE.    *
 *                                                                   *
 *    THE DATA OPTIONS SELECTED FOR THIS EXIT ARE:                   *
 *    EXIT=(DLI2DB2,PATH,DATA,(CASCADE,PATH,DATA,NOKEY)              *
 *-------------------------------------------------------------------*
 *                                                                   *
 *    INPUT FOR THIS PROGRAM : XPCB, XSDB.                           *
 *                                                                   *
 *    OUTPUT:  DISPLAY 'SQLERRM' WHEN SQLERROR OCCURS.               *
 *    UPDATES: UPDATES DB2 TREATMT TABLE                             *
 *                                                                   *
 *           : RETURNS REASON CODE 14 RETURN CODE 16 WHEN PATH   *
 *             NOT SPECIFIED ON THE DBDGEN EXIT STATEMENT,       *
 *             RESULTING IN AN ABEND U3314.                      *
```

```
 *                                                                *
 *----------------------------------------------------------------*
 *    LOGIC:   THIS PROGRAM IS CALLED BY IMS AFTER AN UPDATE TO  *
 *             THE TREATMT SEGMENT AND BEFORE IMS RETURNS TO     *
 *             IMS APPLICATION PROGRAM.                          *
 *                                                               *
 *             THE ADDRESS OF AN XPCB IS PASSED TO THIS PROGRAM  *
 *             FROM IMS. THE XPCB WILL PROVIDE THE ADDRESSES OF  *
 *             THE XSDB FOR DATA, PATH DATA AND BEFORE DATA.     *
 *                                                               *
 *             UPDATE THE DB2 TREATMT TABLE WITH THE UPDATED IMS *
 *             SEGMENT DATA.                                     *
 *                                                               *
 *   HOSPITAL   ***********                                      *
 *   DATA BASE  *         *                                      *
 *              * PATIENT *  KEY FIELD IS PATNO                  *
 *              *         *                                      *
 *              ***********                                      *
 *                   *                                           *
 *                   *                                           *
 *              ***********                                      *
 *              *         *                                      *
 *              * ILLNESS *  KEY FIELD IS ILLDATE                *
 *              *         *                                      *
 *              ***********                                      *
 *                   *                                           *
 *                   *                                           *
 *              ***********  KEY FIELD IS TRTDATE                *
 *              *         *  FIELD, MEDICINE                     *
 *              * TREATMT *  FIELD, QUANTITY                     *
 *              *         *  FIELD, DOCTOR (NOT IN DB2 TABLE)    *
 *              ***********                                      *
 *                                                               *
 *                                                               *
 *      TREATMENT TABLE                                          *
 *                                                               *
 *      ************************************************         *
 *      * PATNUMB * DATEILL * DATETRT * MEDICAT * AMOUNT  *      *
 *      ************************************************         *
 *                                                               *
 * --------------------------------------------------------------*
 */
/* ************************************************************  */
/*                                                               */
/*      E X T E N D E D   D A T A  B A S E  P C B  -- X P C B     */
/*                                                               */
/* ************************************************************ */

DECLARE
  1 XPCB              BASED(XPCB_PTR),
    3 EYECATCHER           CHAR(4),   /* "XPCB" EYECATCHER         */
    3 VERSION              CHAR(2),   /* XPCB VERSION INDICATOR    */
    3 RELEASE              CHAR(2),   /* XPCB RELEASE INDICATOR    */
    3 EXIT_NAME            CHAR(8),   /* SEGMENT EXIT NAME         */
    3 EXIT_RETURN_CODE     FIXED BINARY (15), /* RETURN CODE       */
    3 EXIT_REASON_CODE     FIXED BINARY (15), /* REASON CODE       */
    3 ATABASE_NAME         CHAR(8),   /* PHYSICAL DATA BASE NAME   */
    3 DBD_VERSION_PTR      POINTER,   /* ADDRESS OF DBD VERSION ID */
    3 SEGMENT_NAME         CHAR(8),   /* PHYSICAL SEGMENT NAME     */
    3 CALL_FUNCTION        CHAR(4),   /* CALL FUNCTION             */
    3 PHYSICAL_FUNCTION    CHAR(4),   /* DL/I PHYSICAL FUNCTION    */
    3 FILLER1              FIXED BINARY (31), /* RESERVED          */
    3 DB_PCB_PTR           POINTER,   /* ADDRESS OF DB PCB         */
    3 DB_PCB_NAME          CHAR(8),   /* NAME OF DB PCB            */
    3 INQY_OUTPUT_PTR      POINTER,   /* ADDRESS OF "INQY" OUTPUT  */
    3 IO_PCB_PTR           POINTER,   /* ADDRESS OF I/O PCB        */
    3 ENVIRONMENT-FLAGS    CHAR(1),   /* Environment Flags         */
    /* IMS-ENH-SUPPORT     X'80' RRS SUPPORT AVAILABLE IN SYSTEM */
    /* IMS-RRS-ENABLED     X'40' RRS=Y WAS SPECIFIED             */
    /* CALL_AT_COMMIT      X'20' SET BY EXIT-CALL DURING COMMIT  */
    /* XPCB_LOGX_FORMAT    X'10' REDUCED 9904 FORMAT             */
    /* XPCB_EXIT_WAS_CALLED X'08' INTERNAL FLAG USED BY IMS      */
    /* XPCB_DPROP_EXIT     X'04' SET BY DPROP EXIT ROUTINE       */
    3 NEWFILLER            CHAR(1),           /* Reserved          */
    3 CONC_KEY_LENGTH      FIXED BINARY (15), /* LENGTH OF FULLY   */
                                              /* CONCATENATED KEY FOR SEGM */
    3 CONC_KEY_PTR         POINTER,   /* ADDRESS OF PHYSICAL FULLY */
                                      /* CONCATENATED KEY FOR SEGM */
    3 DATA_XSDB_PTR        POINTER,   /* ADDRESS OF XSDB FOR       */
                                      /* PHYSICAL SEGMENT DATA     */
    3 BEFORE_XSDB_PTR      POINTER,   /* ADDRESS OF XSDB FOR       */
                                      /* PHYSICAL BEFORE DATA      */
```

```
    3 PATH_XSDB_PTR         POINTER,   /* ADDRESS OF XSDB FOR       */
                                       /* PHYSICAL PATH DATA        */
    3 FILLER3               POINTER,   /* RESERVED                  */
    3 FILLER4               POINTER,   /* RESERVED                  */
    3 FILLER5               POINTER,   /* RESERVED                  */
    3 EXIT_WORK_PTR         POINTER,   /* ADDRESS OF 256 BYTE AREA  */
                                       /*        FOR THE EXIT       */
    3 NULL_PTR              POINTER,   /* NULL POINTER VALUE        */
    3 FILLER6               POINTER,   /* RESERVED                  */
    3 CALL_TIMESTAMP        CHAR(8),   /* TIMESTAMP OF CALL         */
    3 FILLER7               POINTER;   /* RESERVED FOR NULLS AT END */
DECLARE XPCB_PTR   POINTER;
/* ************************************************************** */
/*                                                                */
/*      E X T E N D E D   S E G M E N T  D A T A   -- X S D B     */
/*                                                                */
/* ************************************************************** */

DECLARE
  1 XSDB              BASED(XSDB_PTR),
    3 EYECATCHER            CHAR(4),    /* "XSDB" EYECATCHER         */
    3 VERSION               CHAR(2),    /* XSDB VERSION INDICATOR    */
    3 RELEASE               CHAR(2),    /* XSDB RELEASE INDICATOR    */
    3 NEXT_PTR              POINTER,    /* NEXT XSDB POINTER         */
    3 DATABASE_NAME         CHAR(8),    /* PHYSICAL DATA BASE NAME   */
    3 SEGMENT_NAME          CHAR(8),    /* PHYSICAL SEGMENT NAME     */
    3 FILLER1               CHAR(4),    /* RESERVED                  */
    3 SEGMENT_LEVEL FIXED BINARY (15),  /* SEGMENT DATA BASE LEVEL   */
    3 KEY_LENGTH    FIXED BINARY (15),  /* LENGTH OF PHYSICAL KEY    */
    3 KEY_PTR               POINTER,    /* ADDRESS OF PHYSICAL KEY   */
    3 FILLER2       FIXED BINARY (15),  /* RESERVED                  */
    3 SEGMENT_LENGTH FIXED BINARY (15), /* LENGTH OF SEGMENT DATA    */
    3 SEGMENT_DATA_PTR      POINTER,    /* ADDRESS OF SEGMENT DATA   */
    3 FILLER3               POINTER,    /* RESERVED                  */
    3 FILLER4               POINTER,    /* RESERVED                  */
    3 FILLER5               POINTER;    /* RESERVED FOR NULLS AT END */

DECLARE XSDB_PTR            POINTER;
DECLARE
  1 SEGMENT_XSDB    LIKE XSDB  BASED(XPCB.DATA_XSDB_PTR);
DECLARE               /* TREATMENT SEGMENT    */
  1 SEGMENT_DATA   BASED(SEGMENT_XSDB.SEGMENT_DATA_PTR),
    3 SEGMENT_DATA_TRTDATE   CHAR(6),      /* SEGMENT KEY */
    3 SEGMENT_DATA_MEDICINE  CHAR(10),
    3 SEGMENT_DATA_QUANTITY  CHAR(4),
    3 SEGMENT_DATA_DOCTOR    CHAR(10);
DECLARE
  1 BEFORE_XSDB  LIKE XSDB  BASED(XPCB.BEFORE_XSDB_PTR);
DECLARE               /* BEFORE TREATMENT SEGMENT */
  1 BEFORE_DATA  BASED(BEFORE_XSDB.SEGMENT_DATA_PTR),
    3 BEFORE_DATA_TRTDATE   CHAR(6),      /* SEGMENT KEY */
    3 BEFORE_DATA_MEDICINE  CHAR(10),
    3 BEFORE_DATA_QUANTITY  CHAR(4),
    3 BEFORE_DATA_DOCTOR    CHAR(10);
DECLARE
  1 PATH_XSDB    LIKE XSDB  BASED(PATH_XSDB_PTR);
DECLARE               /*  PATIENT  SEGMENT    */
  1 PATH_DATA BASED(PATH_XSDB.SEGMENT_DATA_PTR),
    3 PATHSEG_PATNO          CHAR(5),      /* SEGMENT KEY */
    3 PATHSEG_NAME           CHAR(10),
    3 PATHSEG_ADDR           CHAR(30); DECLARE
  1 PATH2_XSDB    LIKE XSDB  BASED(PATH2_XSDB_PTR);
DECLARE               /*  PATIENT  SEGMENT    */
  1 PATH2_DATA BASED(PATH2_XSDB.SEGMENT_DATA_PTR),
    3 PATH2SEG_ILLDATE        CHAR(6),      /* SEGMENT KEY */
    3 PATH2SEG_ILLNAME        CHAR(10);
DECLARE PATH2_XSDB_PTR            POINTER;
DECLARE               /* TREATMENT TABLE ROW  */
  1 TREATROW    BASED(XPCB.EXIT_WORK_PTR),
    3 COL_PATNUM         CHAR(5),      /* FROM LEVEL 1 KEY */
    3 COL_ILLDATE        CHAR(6),      /* FROM LEVEL 2 KEY */
    3 COL_TRTDATE        CHAR(6),      /* FROM LEVEL 3 KEY */
    3 COL_MEDICINE       CHAR(10),      /* FROM LEVEL 3 */
    3 COL_QUANTITY       CHAR(4);      /* FROM LEVEL 3 */
          EXEC SQL
            INCLUDE SQLCA;
     /* - DB2 TREATMENT TABLE DECLARATION  */

          EXEC SQL
            DECLARE SYSADM.TREATMNT TABLE
              (PATNUMB VARCHAR (5)    NOT NULL,
               DATEILL VARCHAR (6)    NOT NULL,
```

```
                DATETRT VARCHAR (6)    NOT NULL,
                MEDICAT VARCHAR (10)   NOT NULL,
                AMOUNT  VARCHAR (4)    NOT NULL);

DECLARE                                 /* CALL FUNCTIONS */
    INSERT_FUNCTION       CHAR(4) STATIC INIT('ISRT'),
    DELETE_FUNCTION       CHAR(4) STATIC INIT('DLET'),
    REPLACE_FUNCTION      CHAR(4) STATIC INIT('REPL'),
    CASCADE_FUNCTION      CHAR(4) STATIC INIT('CASC');
DECLARE ZERO      FIXED BINARY (31) STATIC
                   INIT(0);
DECLARE SIXTEEN   FIXED BINARY (31) STATIC
                   INIT(16);

   PATH2_XSDB_PTR = PATH_XSDB.NEXT_PTR;
   TREATROW.COL_PATNUM = PATH_DATA.PATHSEG_PATNO;
   TREATROW.COL_ILLDATE = PATH2_DATA.PATH2SEG_ILLDATE;
   TREATROW.COL_TRTDATE = SEGMENT_DATA.SEGMENT_DATA_TRTDATE;
   TREATROW.COL_MEDICINE = SEGMENT_DATA.SEGMENT_DATA_MEDICINE;
   TREATROW.COL_QUANTITY = SEGMENT_DATA.SEGMENT_DATA_QUANTITY;

    EXEC SQL
         WHENEVER SQLWARNING CONTINUE;
    EXEC SQL
         WHENEVER SQLERROR GOTO BADSQL;
    EXEC SQL
         WHENEVER NOT FOUND GOTO BADSQL;
   IF XPCB.PATH_XSDB_PTR = XPCB.NULL_PTR
   THEN DO;
   GOTO BADPATH;   /* PATH NOT SPECIFIED */
   END;    ELSE DO;       /* PRE-SET CODES TO ZERO */
   XPCB.EXIT_RETURN_CODE = ZERO;
   XPCB.EXIT_REASON_CODE = ZERO;
   END;
        /*====================================*/
        /* IF CALLED FOR DELETE OR CASCADE,   */
        /* PERFORM THE DB2 DELETE.            */
        /*====================================*/
   IF XPCB.PHYSICAL_FUNCTION = DELETE_FUNCTION
   THEN DO;

      EXEC SQL
        DELETE FROM SYSADM.TREATMNT
        WHERE PATNUMB = ::TREATROW.COL_PATNUM AND
        DATEILL = ::TREATROW.COL_ILLDATE AND
             DATETRT = ::TREATROW.COL_TRTDATE;
   END;
        /*========================================*/
        /* IF CALLED FOR INSERT, DO DB2 INSERT CALL */
        /*========================================*/
   IF XPCB.CALL_FUNCTION = INSERT_FUNCTION
   THEN DO;
     EXEC SQL
        INSERT INTO SYSADM.TREATMNT
        VALUES(::TREATROW.COL_PATNUM,
               ::TREATROW.COL_ILLDATE,
               ::TREATROW.COL_TRTDATE,
               ::TREATROW.COL_MEDICINE,
               ::TREATROW.COL_QUANTITY);
   END;
        /*====================================*/
        /* IF CALLED FOR REPLACE, UPDATE THE   */
        /* THE DB2 ROW, IF A FIELD DESTINED TO */
        /* THE DB2 DATA BASE HAS BEEN CHANGED. */
        /*====================================*/
   IF XPCB.CALL_FUNCTION = REPLACE_FUNCTION
   THEN DO;    /* REPLACE */
    IF (SEGMENT_DATA.SEGMENT_DATA_MEDICINE ≠
    BEFORE_DATA.BEFORE_DATA_MEDICINE) |
    (SEGMENT_DATA.SEGMENT_DATA_QUANTITY ≠
    BEFORE_DATA.BEFORE_DATA_QUANTITY)
    THEN DO;   /* UPDATE */
      EXEC SQL
      UPDATE SYSADM.TREATMNT
      SET  MEDICAT = ::SEGMENT_DATA.SEGMENT_DATA_MEDICINE,
           AMOUNT  = ::SEGMENT_DATA.SEGMENT_DATA_QUANTITY
      WHERE PATNUMB = ::TREATROW.COL_PATNUM AND
      DATEILL = ::TREATROW.COL_ILLDATE AND
      DATETRT = ::TREATROW.COL_TRTDATE;
    END;       /* OF UPDATE  */
   END;        /* OF REPLACE */
STOP;
```

```
    BADSQL: DO;    DISPLAY(SQLERRM);
    XPCB.EXIT_RETURN_CODE = 16;
    XPCB.EXIT_REASON_CODE = SQLCODE;
    END;
    BADPATH: DO;
    XPCB.EXIT_RETURN_CODE = 16;
    XPCB.EXIT_REASON_CODE = 14;
    END;
 END DLI2DB2B;
```

Sample Extended Program Communication Block (XPCB)

The segment that is passed in a dependent region and has the Data Capture exit routine defined for it can use the XPCB to identify the segment and call functions, provides the address of a work area, and contains additional information that is passed to the Data Capture exit routine.

This topic provides examples of the XPCB in assembler, COBOL, and PL/I.

Subsections:

Assembler

The following code sample is an example of the XPCB in assembler.

```
         SPACE 3
XPCB     DSECT
XPCB_EYECATCHER          DS    CL4     "XPCB" EYECATCHER
XPCB_VERSION             DS    CL2     XPCB VERSION INDICATOR
XPCB_RELEASE             DS    CL2     XPCB RELEASE INDICATOR
XPCB_EXIT_NAME           DS    CL8     SEGMENT EXIT NAME
XPCB_EXIT_RETURN_CODE    DS    H       RETURN CODE FROM EXIT
XPCB_EXIT_REASON_CODE    DS    H       REASON CODE FROM EXIT
XPCB_DATABASE_NAME       DS    CL8     PHYSICAL DATA BASE NAME
XPCB_DBD_VERSION_PTR     DS    A       ADDRESS OF DBD VERSION ID
XPCB_SEGMENT_NAME        DS    CL8     PHYSICAL SEGMENT NAME
XPCB_CALL_FUNCTION       DS    CL4     CALL FUNCTION
XPCB_PHYSICAL_FUNCTION   DS    CL4     PHYSICAL CALL FUNCTION
                         DS    CL4
XPCB_DB_PCB_PTR          DS    A       ADDRESS OF DB PCB
XPCB_DB_PCB_NAME         DS    CL8     NAME OF DB PCB
XPCB_INQY_OUTPUT_PTR     DS    A       ADDRESS OF "INQY" OUTPUT
XPCB_IO_PCB_PTR          DS    A       ADDRESS OF I/O PCB
XPCB_ENVIRONMENT_FLAGS   DS    X       ENVIRONMENT FLAGS
XPCB_IMS_ENH_SUPPORT     EQU   X'80'   RRS SUPPORT IS AVAILABLE IN SYSTEM
XPCB_IMS_RRS_ENABLED     EQU   X'40'   RRS=Y WAS SPECIFIED
XPCB_CALL_AT_COMMIT      EQU   X'20'   SET BY EXIT - CALL DURING COMMIT
XPCB_LOGX_FORMAT         EQU   X'10'   REDUCED 9904 FORMAT
XPCB_EXIT_WAS_CALLED     EQU   X'08'   INTERNAL FLAG USED BY IMS
XPCB_DPROP_EXIT          EQU   X'04'   SET BY DPROP EXIT ROUTINE
                         DS    X       RESERVED
XPCB_CONC_KEY_LENGTH     DS    H       LENGTH OF CONCATENATED KEY
XPCB_CONC_KEY_PTR        DS    A       ADDRESS OF CONCATENATED KEY
XPCB_DATA_XSDB_PTR       DS    A       ADDRESS OF XSDB FOR DATA
XPCB_BEFORE_XSDB_PTR     DS    A       ADDRESS OF XSDB FOR REPL DATA
XPCB_PATH_XSDB_PTR       DS    A       ADDRESS OF XSDB FOR PATH DATA
XPCB_SETS_ROLS_TOKEN     DS    F       TOKEN FOR SETS-ROLS CALL
                         DS    F       RESERVED
                         DS    F       RESERVED
XPCB_EXIT_WORK_PTR       DS    A       ADDRESS OF WORK AREA
XPCB_ZERO_POINTER        DS    A       ZERO ADDRESS
                         DS    F       RESERVED
XPCB_TIMESTAMP           DS    CL8     TIMESTAMP OF CALL
         EJECT
```

COBOL

The following code sample is an example of the XPCB in COBOL.

```
01  XPCB.
    05  EYECATCHER                    PIC X(04).
    05  VERSION                       PIC X(02).
    05  RELEASE-ID                    PIC X(02).
    05  EXIT-NAME                     PIC X(08).
    05  EXIT-RETURN-CODE              PIC 9(04) COMP.
    05  EXIT-REASON-CODE              PIC 9(04) COMP.
    05  DATABASE-NAME                 PIC X(08).
    05  DBD-VERSION-PTR               POINTER.
    05  SEGMENT-NAME                  PIC X(08).
    05  CALL-FUNCTION                 PIC X(04).
    05  PHYSICAL-FUNCTION             PIC X(04).
    05  FILLER                        PIC 9(08) COMP.
    05  DB-PCB-PTR                    POINTER.
    05  DB-PCB-NAME                   PIC X(08).
    05  INQY-OUTPUT-PTR               POINTER.
    05  IO-PCB-PTR                    POINTER.
    05  ENVIRONMENT-FLAGS             PIC X.
      88  IMS-ENH-SUPPORT            VALUE X'80'.
*                         RRS SUPPORT IS AVAILABLE IN SYSTEM
      88  IMS-RRS-ENABLED            VALUE X'40'.
*                         RRS=Y WAS SPECIFIED
      88  CALL_AT_COMMIT             VALUE X'20'.
*                         SET BY EXIT - CALL DURING COMMIT
      88  XPCB_LOGX_FORMAT           VALUE X'10'.
*                         REDUCED 9904 FORMAT
      88  XPCB_EXIT_WAS_CALLED       VALUE X'08'.
*                         INTERNAL FLAG USED BY IMS
      88  XPCB_DPROP_EXIT            VALUE X'04'.
*                         SET BY DPROP EXIT ROUTINE
    05  FILLER                          PIC X.
* RESERVED
    05  CONC-KEY-LENGTH               PIC 9(04) COMP.
    05  CONC-KEY-PTR                  POINTER.
    05  DATA-XSDB-PTR                 POINTER.
    05  BEFORE-XSDB-PTR               POINTER.
    05  PATH-XSDB-PTR                 POINTER.
    05  FILLER                        POINTER.
    05  FILLER                        POINTER.
    05  FILLER                        POINTER.
    05  EXIT-WORK-PTR                 POINTER.
    05  NULL-PTR                      POINTER.
    05  FILLER                        POINTER.
    05  TIMESTAMP                     PIC X(08).
```

PL/I

The following sample is an example of the XPCB in PL/I.

```
DECLARE
  1 XPCB              BASED(XPCB_PTR),
    3 EYECATCHER           CHAR(4),   /* "XPCB" EYECATCHER         */
    3 VERSION              CHAR(2),   /* XPCB VERSION INDICATOR    */
    3 RELEASE              CHAR(2),   /* XPCB RELEASE INDICATOR    */
    3 EXIT_NAME            CHAR(8),   /* SEGMENT EXIT NAME         */
    3 EXIT_RETURN_CODE     FIXED BINARY (15), /* RETURN CODE       */
    3 EXIT_REASON_CODE     FIXED BINARY (15), /* REASON CODE       */
    3 ATABASE_NAME        CHAR(8),   /* PHYSICAL DATA BASE NAME    */
    3 DBD_VERSION_PTR      POINTER,   /* ADDRESS OF DBD VERSION ID */
    3 SEGMENT_NAME         CHAR(8),   /* PHYSICAL SEGMENT NAME     */
    3 CALL_FUNCTION        CHAR(4),   /* CALL FUNCTION             */
    3 PHYSICAL_FUNCTION    CHAR(4),   /* DL/I PHYSICAL FUNCTION    */
    3 FILLER1              FIXED BINARY (31), /* RESERVED          */
    3 DB_PCB_PTR           POINTER,   /* ADDRESS OF DB PCB         */
    3 DB_PCB_NAME          CHAR(8),   /* NAME OF DB PCB            */
    3 INQY_OUTPUT_PTR      POINTER,   /* ADDRESS OF "INQY" OUTPUT  */
    3 IO_PCB_PTR           POINTER,   /* ADDRESS OF I/O PCB        */
    3 ENVIRONMENT-FLAGS    CHAR(1),   /* Environment Flags         */
       /* IMS-ENH-SUPPORT      X'80' RRS SUPPORT AVAILABLE IN SYSTEM   */
       /* IMS-RRS-ENABLED      X'40' RRS=Y WAS SPECIFIED               */
       /* CALL_AT_COMMIT       X'20' SET BY EXIT-CALL DURING COMMIT    */
       /* XPCB_LOGX_FORMAT     X'10' REDUCED 9904 FORMAT               */
       /* XPCB_EXIT_WAS_CALLED X'08' INTERNAL FLAG USED BY IMS         */
       /* XPCB_DPROP_EXIT      X'04' SET BY DPROP EXIT ROUTINE         */
    3 NEWFILLER            CHAR(1),          /* Reserved          */
    3 CONC_KEY_LENGTH      FIXED BINARY (15), /* LENGTH OF FULLY   */
                                             /* CONCATENATED KEY FOR SEGM */
    3 CONC_KEY_PTR         POINTER,   /* ADDRESS OF PHYSICAL FULLY */
```

```
                                        /* CONCATENATED KEY FOR SEGM */
    3 DATA_XSDB_PTR         POINTER,    /* ADDRESS OF XSDB FOR       */
                                        /* PHYSICAL SEGMENT DATA     */
    3 BEFORE_XSDB_PTR       POINTER,    /* ADDRESS OF XSDB FOR       */
                                        /* PHYSICAL BEFORE DATA      */
    3 PATH_XSDB_PTR         POINTER,    /* ADDRESS OF XSDB FOR       */
                                        /* PHYSICAL PATH DATA        */
    3 FILLER3               POINTER,    /* RESERVED                  */
    3 FILLER4               POINTER,    /* RESERVED                  */
    3 FILLER5               POINTER,    /* RESERVED                  */
    3 EXIT_WORK_PTR         POINTER,    /* ADDRESS OF 256 BYTE AREA  */
                                        /*        FOR THE EXIT       */
    3 NULL_PTR              POINTER,    /* NULL POINTER VALUE        */
    3 FILLER6               POINTER,    /* RESERVED                  */
    3 CALL_TIMESTAMP        CHAR(8),    /* TIMESTAMP OF CALL         */
    3 FILLER7               POINTER;    /* RESERVED FOR NULLS AT END */

DECLARE XPCB_PTR   POINTER;
```

Sample Extended Segment Data Block (XSDB)

The segment that is passed in a dependent region and has the Data Capture exit routine defined for it can use the XSDB, which points to the updated data that is passed to the Data Capture exit routine.

This topic provides examples of the XSDB in assembler, COBOL, and PL/I.

Subsections:

Assembler

The following code sample is an example of the XSDB in assembler.

```
         SPACE 3
XSDB     DSECT
XSDB_EYECATCHER          DS    CL4     "XSDB" EYECATCHER
XSDB_VERSION             DS    CL2     XSDB VERSION INDICATOR
XSDB_RELEASE             DS    CL2     XSDB RELEASE INDICATOR
XSDB_NEXT_PTR            DS    A       NEXT XSDB POINTER
XSDB_DATABASE_NAME       DS    CL8     PHYSICAL DATA BASE NAME
XSDB_SEGMENT_NAME        DS    CL8     PHYSICAL SEGMENT NAME
                         DS    CL4     RESERVED
XSDB_SEGMENT_LEVEL       DS    H       SEGMENT DATA BASE LEVEL
XSDB_KEY_LENGTH          DS    H       LENGTH OF PHYSICAL KEY
XSDB_KEY_PTR             DS    A       ADDRESS OF PHYSICAL KEY
XSDB_LP_KEY_LENGTH       DS    H       LENGTH OF LOGICAL PARENT KEY
XSDB_SEGMENT_LENGTH      DS    H       LENGTH OF SEGMENT DATA
XSDB_SEGMENT_DATA_PTR    DS    A       ADDRESS OF SEGMENT DATA
                         DS    F       RESERVED
                         DS    F       RESERVED
```

COBOL

The following code sample is an example of the XSDB in COBOL.

```
01  XSDB
    05  EYECATCHER                   PIC X(4).
    05  VERSION                      PIC X(2).
    05  RELEASE-ID                   PIC X(2).
    05  NEXT-PTR                     POINTER.
    05  DATABASE-NAME                PIC X(8).
    05  SEGMENT-NAME                 PIC X(8).
    05  FILLER                       PIC X(4).
    05  SEGMENT-LEVEL                PIC 9(4) COMP.
    05  KEY-LENGTH                   PIC 9(4) COMP.
    05  KEY-PTR                      POINTER.
    05  LP-KEY-LENGTH                PIC 9(4) COMP.
    05  SEGMENT-LENGTH               PIC 9(4) COMP.
    05  SEGMENT-DATA-PTR             POINTER.
```

```
    05  FILLER                      POINTER.
    05  FILLER                      POINTER.
```

PL/I

The following code sample is an example of the XSDB in PL/I.

```
DECLARE
  1 XSDB              BASED(XSDB_PTR),
    3 EYECATCHER            CHAR(4),    /* "XSDB" EYECATCHER         */
    3 VERSION               CHAR(2),    /* XSDB VERSION INDICATOR    */
    3 RELEASE               CHAR(2),    /* XSDB RELEASE INDICATOR    */
    3 NEXT_PTR              POINTER,    /* NEXT XSDB POINTER         */
    3 DATABASE_NAME         CHAR(8),    /* PHYSICAL DATA BASE NAME   */
    3 SEGMENT_NAME          CHAR(8),    /* PHYSICAL SEGMENT NAME     */
    3 FILLER1               CHAR(4),    /* RESERVED                  */
    3 SEGMENT_LEVEL FIXED BINARY (15),  /* SEGMENT DATA BASE LEVEL   */
    3 KEY_LENGTH    FIXED BINARY (15),  /* LENGTH OF PHYSICAL KEY    */
    3 KEY_PTR               POINTER,    /* ADDRESS OF PHYSICAL KEY   */
    3 LP_KEY_LENGTH FIXED BINARY (15),  /* RESERVED                  */
    3 SEGMENT_LENGTH FIXED BINARY (15), /* LENGTH OF SEGMENT DATA    */
    3 SEGMENT_DATA_PTR      POINTER,    /* ADDRESS OF SEGMENT DATA   */
    3 FILLER3               POINTER,    /* RESERVED                  */
    3 FILLER4               POINTER,    /* RESERVED                  */
    3 FILLER5               POINTER;    /* RESERVED FOR NULLS AT END */

DECLARE XSDB_PTR            POINTER;
```

Data conversion user exit routine (DFSDBUX1)

The purpose of the Data Conversion exit routine (DFSDBUX1) is to provide a method for modifying segment search arguments, the key feedback area, the I/O area, and the status code.

This topic contains Product-sensitive Programming Interface information.

The Data Conversion user exit routine (DFSDBUX1) gets control at the beginning of a DL/I call and at the end of the call. In the exit routine, you can modify segment search arguments, the key feedback area, the I/O area, and the status code.

Restriction: This exit routine gets control only for calls to full-function databases.

Subsections:

About this routine

Attributes of the routine

Regardless of its function, the exit routine must be written in assembler language, C language, COBOL, or PL/I. Routines written in high-level languages running under Language Environment for z/OS are not supported.

Binding the routine

Bind the exit routine DFSDBUX1 with the RENT attribute into an APF-authorized library. This library can be either IMS.SDFSRESL, SYS1.LINKLIB, or any partitioned data set that can be accessed by a JOBLIB or a STEPLIB DD statement for the IMS control, SAS, batch, or CICS region.

Loading the routine

IMS attempts to load the exit routine on the first database call. If the exit routine fails to load, IMS does not attempt to load it again.

Other considerations

A DBD generation is not required for IMS to call the exit routine.

Recommendation: Perform a DBD generation with the DATXEXIT=YES parameter for DBDs that require the exit routine.

If you do *not* specify the DATXEXIT=YES parameter for a DBD, the call analyzer (DFSDLA00) issues a DFS2097I message if the exit routine specifies that it should continue to be called for that DBD. After issuing message DFS2097I, the call analyzer DFSDLA00 dynamically sets the DATXEXIT parameter to YES for the DBD and continues calling the exit routine. The DFS2097I message appears only once per DBD.

If you bind an exit routine and want to prevent it from being called, remove the DFSDBUX1 exit routine from the library in which you edited it.

If exit routine DFSDBUX1 is available to IMS, it is called regardless of the DATXEXIT parameter specification. If the exit routine determines that the exit routine should not be called again for the DBD, the routine returns abend code X'FF' in the SRCHFLAG field in the JCB (SRCHFLAG EQUA JCBWKR55). Abend code X'FF' causes call analyzer DFSDLA00 to dynamically mark the DBD as not requiring the exit routine. In this case, the exit routine is not called again for that DBD for the duration of the execution of this IMS *or* until the DMB is purged from the DMB pool.

If you use exit routine DFSDBUX1, it is loaded and called on each database call. If you do not want to run the DFSDBUX1 exit routine for every database, create a table in the DFSDBUX1 exit routine that includes the names of the databases you want the routine to process every time it is called. When exit routine DFSDBUX1 is called, it checks the table of database names. If a database name is not in that table, the DFSDBUX1 exit routine flags that database with a X'FF' value in the JCB when it first calls it, which indicates that the database is not processed further.

Preloading the exit routine is not necessary. After it is loaded, the exit routine remains loaded until region termination.

Using IMS callable services with this routine

This exit is not eligible to use IMS callable services.

Issuing SVC calls

In an online environment, the exit routine might be running in cross-memory mode. To prevent 0F8 abends, the exit should avoid issuing SVC calls.

Communicating with IMS

IMS uses the general purpose registers and several IMS control blocks to communicate with the DFSDBUX1 exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
0	The characters 'IN' at the start of the DL/I call and the characters 'OUT' at the end of the DL/I call.
1	Address of the Partition Specification Table.
3	Address of the Database Program Communication Block (DBPCB).
5	Address of the PSB Directory (PDIR).
6	Address of the System Contents Directory (SCD).
7	Address of the Program Specification Block (PSB).
9	Address of the Job Control Block (JCB).
10	Address of the Segment Descriptor Block (SDB).

Register	Contents
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of registers on exit

Before returning to IMS, the exit routine must restore registers 0 through 14. The value of Register 15 must be a 2-byte or less positive value set as follows:

Register	Contents
0	The exit routine has successfully processed the request.
non-0	The exit routine has set a status code or pseudoabend.

Data security and integrity

The exit routine is an extension of the application program with the same capabilities as the application program; the exit routine and the application have equal authorization and limitations.

In batch, the data and the exit routine operate in unprotected key-8 storage. Online, the data and the exit routine operate in unprotected key-7 storage. The exit routine is able to modify data or control blocks that can affect the successful operation of the application program.

Data Entry Database Partition Selection exit routine (DBFPSE00)

Use the Data Entry Database (DEDB) Partition Selection exit routine to partition data for HISAM or SHISAM secondary index databases.

Subsections:

- "About this routine" on page 83
- "Communicating with IMS" on page 85

About this routine

The DEDB Partition Selection exit routine is defined in the primary DEDB database DBD when its secondary index databases are HISAM or SHISAM databases and user partitioning is required.

A partitioned database contains a range of secondary index keys. The DEDB Partition Selection exit routine selects an appropriate partition based on the key value of a search key of the secondary index or other user defined partition selection criteria. The sample partition selection exit routine DBFPSE00 uses the high key for each partition to determine partition selection. A DEDB Partition Selection exit routine can have its own user partition selection criteria.

The PSELRTN= parameter on a XDFLD statement defines a DEDB Partition Selection exit routine for HISAM or SHISAM secondary index databases.

A logical HISAM or SHISAM partition index database can include one or multiple partitions. The PSELOPT=MULT|SNGL parameter on either a PCB statement with the PROCSEQD= parameter, or on a XDFLD statement, determines how partitions are grouped in the index database.

The following naming rules apply to the DEDB Partition Selection exit routine:

- The exit routine name cannot be longer than 8 characters.
- The first character must be alphabetic.
- The remaining characters must be alphabetic, numeric, or #, @, $.

If the PSELRTN= parameter specifies a DEDB Partition Selection exit routine name that violates one or more naming rules, the DBDGEN utility terminates with a MNOTE 8 and message XDFLD235.

The DEDB Partition Selection exit routine supports three functions:

- PTDBINIT: Initialization
- PTDBPSEL: Partition database selection
- PTDBTERM: Termination

The PTDBINIT function is driven when a primary DEDB database that has a DEDB Partition Selection exit routine defined in the PSELRTN= parameter on a XDFLD statement is opened.

The PTDBPSEL function is driven when a primary DEDB database is being accessed or updated using its HISAM or SHISAM secondary index and user partitioning is requested as defined in the primary DEDB database DBD. The DEDB Partition Selection exit routine allows you to select a user partition database among a group of HISAM secondary index databases or a group of SHIASM secondary databases defined in the NAME= parameter on the LCHILD statement and its corresponding XDFLD statement has the DEDB Partition Selection exit routine defined in the PSELRTN= parameter in the primary DEDB database DBD.

The PTDBTERM function is driven when a primary DEDB database that has a DEDB Partition Selection exit routine defined in the PSELRTN= parameter on a XDFLD statement is closed. A DEDB Partition Selection exit routine has similar attributes as a DEDB randomizing module. Table 22 on page 84 summarizes the attributes of a DEDB Partition Selection exit routine for HISAM or SHISAM secondary index databases.

Option to access user partition databases in a user partition group as a separate logical database

Each user partition database can be accessed as a separate database. In addition, all user partition databases in a user partition group can be accessed as a separate logical database using PSELRTN and PSELOPT=MULT|SNGL parameters.

When ACCESS=DB is specified or defaulted on a PCB statement with the PROCSEQD parameter, the user partition databases in a user partition group are accessed as a secondary index to access the primary DEDB database in an alternate sequence.

When ACCESS=INDEX is specified on a PCB statement with the PROCSEQD parameter, the user partition databases in a user partition group are accessed as one single separate logical database. The PSELRTN and PSELOPT=MULT|SNGL are used to control which partition database to access, and one or more subsequent partition databases are in the separate logical database.

The SENSEGS statements in a PCB with the PROCSEQD parameter for both ACCESS=DB and ACCESS=INDEX are the same even though the primary DEDB database is not accessed when ACCESS=INDEX is specified. This requirement allows compatibility of PSBGEN utility and ACBGEN utility for ACCESS=DB and ACCESS=INDEX.

Attributes of the routine

The following table shows the attributes of the Data Entry Database Partition Selection exit routine.

Table 22. Data Entry Database Partition Selection exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	The name given to the load module used for partition selection should also appear in the DBD generation associated with the database. The load module name must be the value of the "mod" parameter of the PSELRTN= parameter on the XDFLD statement in the DEDB DBD generation.
Link editing	After you compile and test your routine, bind it into IMS.SDFSRESL, SYS1.LINKLIB, or any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control and SAS regions.

Table 22. Data Entry Database Partition Selection exit routine attributes (continued)

Attribute	Description
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSRC (member name DBFPSE00).

Loading and deleting the routine

One DEDB Partition Selection exit routine can be shared by both HISAM and SHISAM secondary index databases. A DEDB Partition Selection exit routine resides in the IMS.SDFSRESL, SYS1.LINKLIB, or any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control and SAS regions.

When a primary DEDB database has a DEDB Partition Selection exit routine defined in the PSELRTN= parameter, IMS loads the exit at IMS initialization or at **/START DB** or **UPDATE DB START(ACCESS)** command if the exit has not been loaded.

When a primary DEDB database is closed, its DEDB Partition Selection exit routine is logically deleted. When all the primary DEDB databases sharing the DEDB Partition Selection exit routine are closed, the DEDB Partition Selection exit routine is physically deleted.

When a DEDB Partition Selection exit routine is physically deleted in an IMS system, you can refresh your DEDB Partition Selection exit routine if you need to update your exit routine. After you have refreshed your DEDB Partition Selection exit routine in the library where it resides, issue a **/STA DB** or **UPDATE DB START(ACCESS)** command on the primary DEDB database to load the updated DEDB Partition Selection exit routine.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list mapped by DBFPTDBP macro
13	Address of save area chain for use by this routine.
14	Return address of IMS.
15	Entry point of exit routine.

Contents of registers on exit

Before returning to IMS, the routine must restore all registers except for register 15, which must contain one of the following:

Return code	Meaning
0	Successful completion
4	Unsuccessful. If the exit function was initialization, this return code indicates that the primary DEDB is marked unavailable for access. If the exit function was termination, the primary DEDB is unaffected.

DEDB Partition Selection parameter list

The following table describes the parameter list for the DEDB Partition Selection exit routine (mapped by DBFPTDBP). The DBFPTDBP parameter list macro is located in the IMS macro target library SDFSMAC.

Related concepts

DEDB partitioned secondary indexes (Database Administration)

Sample data entry database randomizing routines (DBFHDC40 / DBFHDC20 DBFHDC44 / DBFHDC24 DBFHDC2S)

A data entry database randomizing module is required for placing root segments in or retrieving them from a DEDB.

Subsections:

About this routine

Several DEDBs can share the same routine, but all areas in a DEDB must use the same routine.

If you are using data sharing, you must use the same randomizing routine on both systems.

IMS supplies sample DEDB randomizing modules (DBFHDC40, DBFHDC20, DBFHDC44, and DBFHDC24) on IMS.ADFSSRC. You can use one of these IMS-supplied routines or you can write your own.

Sample randomizer module DBFHDC2S is in IMS.ADFSSMPL data set. You must understand the key structure of the database and modify this sample appropriately before you use it.

DBFHDC20 and DBFHDC24 are limited two-stage randomizers that are intended for use with the DEDB Alter Utility.

DBFHDC20 is a two-stage randomizer that is based on DBFHDC40, and DBFHDC24 is a two-stage randomizer that is based on DBFHDC44. They have the same attribute and interface as DBFHDC40 and DBFHDC44.

These randomizers first hash the root key to an area by using an arbitrary 4K RAPs/area, and then re-hash the key within the selected area by using the standard DBFHDC4x technique. As a result, a key will not move between areas even if the total number of RAPs in the DEDB changes as a result of changes to the ROOT or UOW parameters for any particular areas in the DBD.

However, if the number of areas that are defined in the DBD changes, a key might move between areas across the DBD change. In that sense, they are *limited* two-stage randomizers because a true two-stage randomizer would not move a key between areas even if the number of areas that are defined in the DBD changes. Such randomizers are usually table-driven and require a detailed knowledge of the key structure and key frequency distribution.

For purposes of the DEDB Alter utility, a DEDB that uses DBFHDC20 or DBFHDC24 can be the target of a DBD alteration that enlarges or reduces the UOW or ROOT parameters of individual areas, so long as the number of areas in DBD does not change.

Restrictions: When you first convert from DBFHDC40 or DBFHDC44, a full unload and reload of the DEDB is required because DBFHDC20 and DBFHDC24 will not randomize keys to the same area or RAP as DBFHDC40 or DBFHDC44.

DBD can be changed to specify RMNAME=(DBFHDC20,2) or (DBFHDC24,2). However, the restriction on changing the number of areas must be obeyed because it is not enforced by IMS. Adding or deleting an area is a database-level change for purposes of online change, even with a randomizer that is defined as "two stage". If the DBD is changed to add or delete an area, a full unload and reload of the DEDB is required.

The following table shows the attributes of the Data Entry Database Randomizing routine.

Table 23. Data Entry Database randomizing routine attributes

Attribute	Description
IMS environments	DB/TM, DBCTL
Naming convention	The name given to the load module used for randomizing functions with a specific database should also appear in the DBD generation that is associated with the database. The load module name must be the value of the "mod" parameter of the RMNAME= operand on the DBD statement in the DEDB DBD generation.
Binding	After you compile and test your randomizing module, bind it into IMS.SDFSRESL, SYS1.LINKLIB, or any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control and SAS regions.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	IMS.ADFSSRC (member name DBFHDC40) IMS.ADFSSRC (member name DBFHDC20) IMS.ADFSSRC (member name DBFHDC44) IMS.ADFSSRC (member name DBFHDC24) IMS.ADFSSMPL (member name DBFHDC2S)

Loading the routine

All randomizing modules are loaded from their resident library by IMS. The name of the module is the name that you specified in the RMNAME parameter of the DBD statement of the database description (DBD).

Related Reading: For details on coding the RMNAME parameter, see *IMS Version 15.2 System Utilities*.

You can use one copy of the randomizing module to service several databases that are concurrently open. At initialization time, the randomizing module can be placed in the main storage or the LPA (link pack area). When running under z/OS, the randomizing module is loaded into the Common Service Area (CSA). If you are to bind with RMODE ANY, you can load it into the Extended Common Service Area (ECSA).

Activating the routine

When an application program issues a Get Unique or Insert call that operates on a root segment of a DEDB database, the user-supplied randomizing module is activated.

The source of the root key that IMS supplies to the randomizing routine is as follows:

- For a root insert, it is taken from the I/O area containing the root to be inserted.
- For a call qualified on the root key, it is the key value in the segment search argument.

Related Reading: For information about processing Get Next (GN) calls qualified on the root key and calls with root qualification that allow a range of key values, see *IMS Version 15.2 Application Programming*.

The key is supplied to the randomizing module for conversion to a relative block number and anchor point number within the database. In addition to the key supplied by an application program, parameters from the DBD generation associated with the database being used are available to the randomizing module.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of registers on entry

On entry to the randomizing module, the registers contain the following:

Register	Contents
0	Number of entries in the MRMB (total number of areas in the DEDB).

Register	Contents
1	Address of first MRMB the routine uses.
2	Size of an entry in the MRMB.
3	Address of the root key.
4	Length of the root key in bytes.
5	Total number of RAPs in the DEDB.
6	Address of an eight-word area that the randomizing module can use.
10	Address of the EPST (Extended Partition Specification Table).
11	Address of the ESCD (Extended System Content Directory).
13	Address of save area. The routine must not change the first three words
14	Return address to IMS.
15	Entry point of randomizing module.

The randomizing module must neither change the key value nor modify any control blocks.

Note: When you run z/OS batch utilities (such as DBFUCDX0 or MSDB-to-DEDB conversion), register 10 contains decimal -1 (X'FFFFFFFF') and register 11 contains zeros. Specific utilities might have additional communication requirements.

Description of parameters

MRMB

To support the facility of randomizing within an area, the routine is passed the address of a Randomizing Module Block (MRMB).

Each area has one 3-word entry. MRMB entries are built in the same order as their associated AREA macros in the DBDGEN for the database. The content of an entry is mapped by DBFMRMB macro and contains the following:

```
MRMB      DSECT
MRMBARTD      DS      0F   START IS WORD-ALIGNED
MRMBARTC      DS      F    ADDRESS OF THE AREA SELECTED
MRMBARTI      DS      F    NUMBER OF ANCHOR POINTS IN THIS AREA
MRMBARTN      DS      F    CUMULATIVE NUMBER OF ANCHOR POINTS
*            IN ALL AREAS OF THE DEDB UP TO AND
*            INCLUDING THIS ONE
MRMBARTZ      DS      0F   END OF THIS ENTRY, START OF NEXT
MRMBARTL      EQU     MBMBARTZ-MRMBARTD
*            LENGTH OF A SINGLE ENTRY
```

Caller Environment

This field contains 4-byte characters to allow the XCI randomizer to distinguish between the IMS online or OS batch caller. The value 'IMS ' indicates IMS online caller, and the value 'OS ' indicates OS batch caller.

Contents of registers on exit

Before returning to IMS, your routine must restore all registers, except for registers 0, 1, and 15, which must contain the following:

Register	Contents
0	Relative root anchor point number within the selected area (0 for first root anchor point).
1	DMAC address of the area selected.
15	Return code interpreted as follows:

Register	Contents	
	Return code	**Meaning**
	0	Register 1 contains the address of the area selected. If the area is not contained in the DMCB or the HSSP sublist, ABENDU1021 is issued.
	4	Status 'FM' needs to be issued.
	Any other return code causes ABENDU1021 to be issued.	

When randomizing through the entire DEDB, the randomizing module must derive an area and a relative root anchor point number to conform to the exit interface. You can use the third word of the MRMB entry to accomplish this.

Related concepts

"Guidelines for writing IMS exit routines" on page 3
Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Database Description (DBD) Generation utility (System Utilities)

Sample DEDB randomizing routines (DBFHDC40)

You can use the IMS-supplied sample DEDB randomizing modules DBFHDC40 on IMS.ADFSSRC.

The sample exit routine is based on the generalized Randomizing Routine (DFSHDC40) and has been modified to work with DEDB databases. The modifications are:

1. The module uses the DEDB input and output interfaces.
2. The module can return an anchor point in block 1, because DEDB areas do not use a bit map at this location.

Extended call interface (XCI) option

The XCI option specifies that this DEDB uses the extended call interface when making calls to the randomizer.

The extended call interface (XCI) option can be specified in the RMNAME= parameter list in the DBD statement of a DBDGEN.

Subsections:

About this routine

The XCI option specifies that this DEDB uses the extended call interface when making calls to the randomizer. This option allows the XCI randomizer to be called in 3 different ways. On initialization of IMS, or during a /START DB command, IMS will first load the randomizer and then make an 'INIT' call to the randomizer to invoke its initialization routines. During a /DBR DB command, IMS will make a 'TERM' call to the randomizer to invoke the termination routines before unloading the randomizer. The normal randomizing call is made when the application issues a GU or ISRT call on a root segment. The XCI randomizer option is valid only for DEDBs.

Attributes of the routine

The attributes of the routine are the same as the non-XCI randomizer.

Invoking the routine

An XCI randomizer is invoked with an initialization call during Fast Path initialization and during a /START DB command. The XCI randomizer is invoked with a termination call during a /DBR DB command. Otherwise, a regular randomizing call is made to the XCI randomizer when an application program issues a GU or ISRT call which operates on a root segment of a DEDB database, just as in a non-XCI randomizer.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Note: In an OS batch caller environment, you can set the values of the IMS name and ECB address fields to zeros. These fields are normally used for randomizing, initialization, and termination calls, but are not used in an OS batch caller environment.

Contents of registers on entry for a randomizing call

On entry for a randomizing call, register 0 contains the constant 'XCI ' (be sure to include a space after the 'XCI').

Register 1 contains the address of the parameter list with the following layout.

Table 24. Sample Parameter List for a Randomizing Call

Hex Offset	Contents
X'0'	0
X'4'	Number of areas
X'8'	Address of randomizing module block (MRMB)
X'C'	Size of MRMB
X'10'	Address of key
X'14'	Key length
X'18'	Total number of route anchor points (RAPs)
X'1C'	Address of work area
X'20'	Any user data
X'24'	0 (XCI parameter version field)
X'28'	8-byte IMS name with trailing blanks
X'30'	IMS level, specified as the value of the &DFSLEV variable of the DFSLEV macro
X'34'	8-byte PSB name with trailing blanks
X'3C'	8-byte caller environment label with trailing blanks: IMS for an online IMS caller or OS for an OS batch caller

Contents of registers on entry for an initialization call

On entry for an initialization call, register 0 contains the constant 'XCI ' (be sure to include a space after the 'XCI').

Register 1 contains the address of the parameter list with the following layout.

Table 25. Sample Parameter List for an Initialization Call

Hex Offset	Contents
X'0'	4
X'4'	Address of the DEDB master control block (DMCB)
X'8'	Address of an event control block (ECB)
X'B'	Any user data
X'10'	0 (XCI parameter version field)
X'14'	8-byte IMS name with trailing blanks
X'1C'	IMS level, specified as the value of the &DFSLEV variable of the DFSLEV macro
X'20'	8-byte caller environment label with trailing blanks: IMS for an online IMS caller or OS for an OS batch caller

Contents of registers on entry for a termination call

On entry for a termination call, register 0 contains the constant 'XCI ' (be sure to include a space after the 'XCI').

Register 1 contains the address of the parameter list with the following layout.

Table 26. Sample Parameter List for a Termination Call

Hex Offset	Contents
X'0'	8
X'4'	Address of the DEDB master control block (DMCB)
X'8'	Address of an event control block (ECB)
X'B'	Any user data
X'10'	0 (XCI parameter version field)
X'14'	8-byte IMS name with trailing blanks
X'1C'	IMS level, specified as the value of the &DFSLEV variable of the DFSLEV macro
X'20'	8-byte caller environment label with trailing blanks: IMS for an online IMS caller or OS for an OS batch caller

XCI Parameter Version Field

The content of the XCI parameter version field is determined by the version of IMS that is using the XCI randomizer.

If the XCI randomizer runs on multiple versions of IMS, you must check the XCI version number. The version number will be incremented when new fields are added. Before accessing fields that are added with a new version number, the version must be checked to ensure that the fields exist.

Contents of registers on exit from a randomizing call

The contents of registers on exit from a randomizing call are as follows:

Register	Contents
0	Relative root anchor point number within the selected AREA (0 for first root anchor point).

Register	Contents	
1	DMAC address of the AREA selected.	
15	Return code interpreted as follows:	
	Return Code	**Meaning**
	0	Register 1 contains the address of the area selected. If the area is not contained in the DMCB or the HSSP sublist, ABENDU1021 is issued.
	4	Status 'FM' needs to be issued.
	Any other return code causes ABENDU1021 to be issued.	

Contents of registers on exit from an initialization call

Register	Contents
1	Reason code for a non-zero return code.
15	Return code.

Contents of registers on exit from a termination call

The contents of registers on exit from a termination call are as follows:

Register	Contents
1	Reason code for a non-zero return code.
15	Return code.

Data Entry Database Resource Name hash routine (DBFLHSH0)

The IMS DEDB Resource Name hash routine is used with the Internal Resource Lock Manager (IRLM) and enables IMS and DBCTL to maintain and retrieve information about the control intervals (CIs) used by sharing subsystems.

Subsections:

About this routine

The routine performs a hashing function on the high-order three bytes of the relative byte address (RBA) representing a CI and uses the hashing result as a displacement into the hash table. If you are using IRLM in your system, the routine IMS supplies (DBFLHSH0) or the replacement routine that you write yourself is called automatically.

You can write the routine and bind it as *reentrant* (RENT) like the one supplied by IMS. It receives control and must return control in 31-bit addressing mode. It must be able to execute in cross-memory and TASK modes.

Important: All IMS systems sharing data must use the same hashing routine or the contents of DEDBs might be lost. IMS does not check to ensure that the routines are the same.

Attributes of the routine

The following table shows the attributes of the Data Entry Database Resource Name Hash routine.

Table 27. Data Entry Database resource name hash routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL
Naming convention	You must name this exit routine DBFLHSH0.
Binding	After you compile and test the routine, bind it into IMS.SDFSRESL or to the library specified in the USERLIB= parameter of the IMSGEN macro statement.
Including the routine	At system definition time, you must specify the name of your routine in the UHASH parameter of the DBC, FDR, or IMS procedure. **Related Reading:** For details, see the on the UHASH and the above procedures in *IMS Version 15.2 System Definition*.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSMPL (member name DBFLHSH0)

Assembling the routine

In a multiple-IMS environment, all IMS systems must use the same hashing routine and compile that routine at the same time. If you write your own routine, you must store the compile time in the module using &SYSDATE and &SYSTIME. You also must place the address of the date and time in the first field of the routine's CSECT.

Communicating with IMS

IMS uses the entry registers and parameter list, and the exit registers to communicate with the routine.

Contents of registers on entry

On entry, the routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of Extended Partition Specification Table (EPST).
13	Address of save area. The routine must not change the first three words.
14	Return address to IMS.
15	Entry point of hash routine.

Description of parameters

As input to the hashing routine, you need to supply *one* of the following:

- the high-order byte of an RBA.
- the names of both a database and an area.

The routine performs an EXCLUSIVELY OR on this input, stores it in a field, and returns a hash value result to the field EPSTRSHS.

EPST (Extended Program Specification Table) input to the routine

Register 1 points to the extended program specification table (EPST) that contains this input as follows:

Field name	Content
EPSTRSHS	Hashing routine result. Only the low-order 14 bits are significant.

Field name	Content
EPSTRSID	Start of the lock name to be hashed. Lock resource name consists the following are shown in the following list: **EPSTLKID** A lock identifier. If EPSTLKID = 0, resource name is for CI. If EPSTLKID is not zero, name is for the area. 1 byte. See the following figure. **EPSTRBA** Bit 0 through 23 of RBA. 3 bytes. **EPSTDMCB** DB Number as defined by DBRC. 2 bytes. **EPSTAREA** Area number. 1 byte. **EPSTDBNM** Database name. 8 bytes. **EPSTARNM** Area name. 8 bytes.

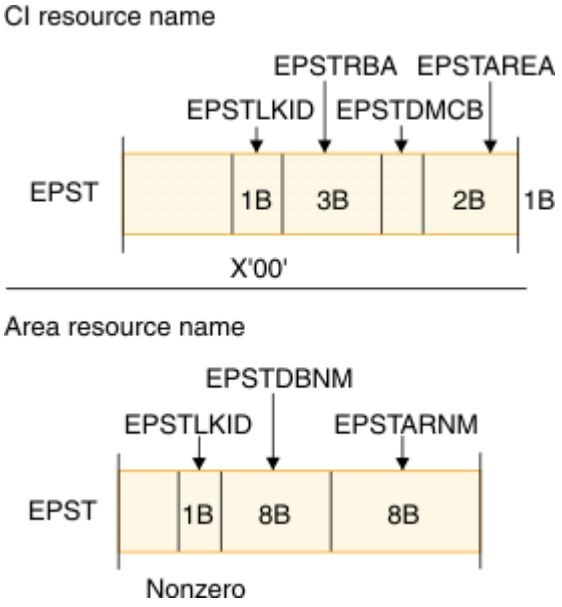


Figure 3. Lock resource name

EPST DSECT

The DSECT of the extended program specification table (EPST) (name: DBFEPST), and the DEDB area control list (DMAC) (name: DBFDMAC) can be used. The DMAC address is set at the EPSTDMAA field.

Related concepts

"Guidelines for writing IMS exit routines" on page 3
Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

Resource name hash routine (Database Administration)

Sample hashing routine result format

Be aware that the IMS-supplied sample hashing routine (DBFLHSH0) has a particular layout and organization for the segments it contains.

The following figure shows the layout of the hash value stored in EPSTRSHS using the IMS-supplied routine DBFLHSH0.

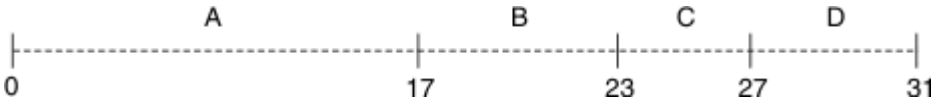


Figure 4. Format of a hash value

The following table describes the segments within a hash value and their sizes.

Table 28. Segments of a hash value

Segment	Description	Size
A	Bits 0 - 17 of EPSTRSHS	18 bits
B	Bits 21 - 25 of CI RBN XOR'd COMB value	5 bits
C	Bits 26 - 29 of CI RBN [1]	4 bits
D	Bits 16 - 20 of CI RBN XOR'd COMB value [2]	5 bits

Note:

1. COMB VALUE (bits 3 - 7) = bits 11 - 15 of DMCB XOR'd with bits 7, 6, 5, 4, and 3 of the area number.
2. CI RBN = RBA divided by the CI size.

Data Entry Database Sequential Dependent Scan utility exit routine (DBFUMSE1)

You can write an exit routine that is used with the DEDB Sequential Dependent Scan utility to copy and process a subset of the segments that are scanned by the utility.

Subsections:

- "About this routine" on page 95
- "Communicating with IMS" on page 96

About this routine

The DEDB Sequential Dependent Scan utility might change both the content and length of the segments scanned. You can choose to sort or not to sort the segments.

If you do not write an exit routine, the Scan utility defaults to passing unchanged segment contents through the range you have specified for scanning. If you do not specify a limit on the range of segments that the utility can scan, the utility scans and copies all of the dependent segments.

Indoubt segments are not passed to this exit routine.

Related Reading: For guidance-level information to help you determine whether to write an exit routine for use with the Scan utility, see *IMS Version 15.2 Database Utilities*.

You can write the routine and bind it as *reentrant* (RENT) like the one supplied by IMS. The routine receives control and must return control in 31-bit addressing. The routine must be able to execute in cross-memory and TASK modes.

Attributes of the routine

The following table shows the attributes of the Data Entry Database Sequential Dependent Scan Utility exit routine.

Table 29. Data Entry Database sequential dependent scan utility exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	This exit routine has no specific naming requirements or restrictions; standard naming conventions apply.
Link editing	After you compile your routine, include it into IMS.SDFSRESL or into any operating system partitioned data set to which access is provided with a JOBLIB or STEPLIB control region JCL statement.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine is not eligible to use IMS callable services.

Calling the routine

If you want IMS to call your routine instead of the IMS-supplied routine (DBFUMSE0), you must specify the name of your routine in the EXIT control statement of the SYSIN DD data set of the Scan Utility JCL.

Related Reading: For details, see *IMS Version 15.2 Database Utilities*.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list. The parameter list is mapped by macro DBFUTDW.
13	Address of save area. The exit routine must not change the first three words.
14	Return address of IMS.
15	Entry point of exit routine.

Contents of registers on exit

Before returning to IMS, the routine must restore all registers except for register 15, which must contain one of the following:

Return code	Meaning
0	Use segment.
4	Do not use segment.

Related concepts

“Guidelines for writing IMS exit routines” on page 3
Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

Related reference

“Exit routine naming conventions” on page 3

Each routine name should adhere to naming conventions, including both standard z/OS conventions, and conventions that are specific to the routine.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample DEDB Sequential Dependent Scan utility exit routine (DBFUMSE1)

The sample DEDB sequential dependent scan utility exit routine is an example showing entry and exit code to help you write your own scan routine.

The following code sample is not a usable exit routine provided by IMS nor is it found in IMS.SDFSSMPL library.

```
   TITLE 'DBFUMSE1 IMS DEDB ONLINE UTILITY SCAN EXIT'
*************************************************************************
*                                                                       *
*  MODULE NAME : DBFUMSE1                                               *
*                                                                       *
*  TITLE : STANDARD EXIT FROM SCAN UTILITY                              *
*                                                                       *
* CONTAINS RESTRICTED MATERIALS OF IBM                                  *
*  COPYRIGHT : REFERENCE MODULE DBFCOPYR                                *
*                                                                       *
*  ENTRY POINT(S)/PURPOSE : DBFUMSE1                                    *
*                                                                       *
*  FUNCTION : THIS IS A SAMPLE OF THE SCAN UTILITY USER EXIT.           *
*             ITS PURPOSE IS TO DEFINE THE INTERFACE BETWEEN            *
*             THE UTILITY AND THE EXIT.  IT IS NOT INTENDED TO          *
*             BE A USABLE EXIT.  IN THIS EXAMPLE, OUTPUT TO THE         *
*             SCAN DATASET IS SUPPRESSED IF THE SEGMENT BEGINS          *
*             WITH HEX ZEROES.                                          *
*                                                                       *
*  ENTRY INTERFACES:                                                    *
*                                                                       *
*   REGISTERS AT ENTRY : R1 ADDRESS OF USER PARAMETER LIST              *
*                        R13 ADDRESS OF SAVE AREA                       *
*                        R14 ADDRESS OF RETURN POINT                    *
*                        R15 ADDRESS OF ENTRY POINT                     *
*                        REGISTERS ARE SAVED AND RESTORED BY THE        *
*                            CALLING MODULES.                           *
*                                                                       *
*   CONTENT OF PARAMETER LIST (UTDWUSER) :                              *
*             UTDWDATA - ADDRESS OF SEGMENT (FULLWORD)                  *
*                                ZERO AFTER LAST SEGMENT                *
*                                1. AT ENTRY ADDRESS OF SEGMENT         *
*                                2. AT EXIT ADDRESS OF DATA TO BE       *
*                                   PICKED UP AND PUT INTO SCAN         *
*                                   OUTPUT DATA SET REFERRED TO         *
*                                   BY SCANCOPY DD CARD.                *
*             UTDWMIN -  MINIMUM LENGTH OF SEGMENT (HALFWORD)           *
*                                AS IN DBD-GENERATION                   *
*             UTDWMAX -  MAXIMUM LENGTH OF SEGMENT (HALFWORD)           *
*                                AS IN DBD-GENERATION                   *
*             UTDWUFLD - FIELD FOR USER (FULLWORD)                      *
*                                ZERO WITH FIRST SEGMENT,               *
*                                UNCHANGED BY THE UTILITY               *
*             UTDWMOUT - MAXIMUM SEGMENT LENGTH (HALFWORD)              *
*                                                                       *
*                        NOTE: THE USER MAY CHANGE LENGTH AND           *
*                                CONTENT OF THE SEGMENT USING HIS       *
*                                OWN WORKSPACE. IF HOWEVER THE LENGTH*
*                                EXCEEDS THE LENGTH OF THE SCAN         *
*                                OUTPUT BUFFER - 8 THE UTILITY IS       *
*                                TERMINATED.                            *
*                                                                       *
*   DATA/OTHER : NONE                                                   *
*                                                                       *
*  EXIT INTERFACES :                                                    *
*   REGISTERS AT EXIT : R15 CONTAINS RETURN CODE                        *
*   RETURN CODES :   00  USE SEGMENT                                    *
*                    04  DO NOT USE SEGMENT                            *
*                                                                       *
*    DATA/OTHER :  NONE                                                 *
*                                                                       *
*  EXTERNAL ROUTINES CALLED : NONE                                      *
```

```
*                                                                     *
*  TABLES/WORKAREAS : NONE                                            *
*                                                                     *
*  REGISTER USAGE : R1  PARAMETER LIST                                *
*                   R2  SEGMENT ADDRESS                               *
*                   R12 MODULE BASE REGISTER                          *
*                   R14 RETURN ADDRESS                                *
*                   R15 RETURN CODE - 00  WRITE SEGMENT               *
*                                     04  DO NOT WRITE SEGMENT        *
*                                                                     *
*  MESSAGE NUMBERS :  NONE                                            *
*                                                                     *
*  ABEND CODES : NONE                                                 *
*                                                                     *
***********************************************************************
         EJECT ,
*PCODE:
***********************************************************************
*                                                                     *
*     IF SEGMENT EXISTS                                               *
*        IF THE SEGMENT STARTS WITH X'00'S                            *
*           SET RC=4  (DON'T WRITE THE SEGMENT)                       *
*        ELSE                                                         *
*           SET RC=0  (WRITE THE SEGMENT)                             *
*        ENDIF                                                        *
*     ELSE                                                            *
*        SET RC=4  (DON'T WRITE THE SEGMENT)                          *
*     ENDIF                                                           *
*     RETURN                                                          *
*                                                                     *
***********************************************************************
*ENDPCODE:
         SPACE 10
         PRINT NOGEN
         REQUATE
         DBFUTDW                   DSECT FOR PARM LIST
         SPACE 10
DBFUMSE1 CSECT
         USING DBFUMSE1,R12        MODULE BASE REGISTER
         USING UTDWUSER,R1         PARAMETER LIST BASE REGISTER
         L     R2,UTDWDATA         GET ADDRESS OF SEGMENT
         LTR   R2,R2               IS THERE A SEGMENT?
         BZ    NOWRITE             NO SEGMENT, DON'T WRITE
*
        LA    R2,2(,R2)           SKIP PAST SEGMENT LENGTH
         CLC   0(6,R2),ZEROES       DOES SEGMENT START WITH 0'S?
         BNE   WRITESEG             NON-ZERO DATA.  WRITE IT.
*                                   OTHERWISE, DON'T WRITE IT.
NOWRITE  DS    0H
         LA    R15,4
         BR    R14
WRITESEG DS    0H
         XR    R15,R15
         BR    R14
ZEROES   DC    XL6'00'
         END
```

HALDB Partition Selection exit routine (DFSPSE00)

You can develop a HALDB Partition Selection exit routine so that PHDAM, PHIDAM, or PSINDEX databases can select partitions by criteria other than high key.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

You can specify the name of the HALDB Partition Selection exit routine during DBD generation, with the HALDB Partition Definition utility, or on the DBRC INIT.DB command.

Use one of the following options to specify the name of the exit routine:

- During DBD generation, use the PSNAME keyword.
- With the HALDB Partition Definition utility, specify the exit routine name as the Partition Selection name.
- Use the PARTSEL keyword on the DBRC INIT.DB command when you register a HALDB database with DBRC.

If you do not specify an exit routine, IMS selects a partition using the high key method and does not invoke the HALDB Partition Selection exit routine.

The following table shows the attributes of the HALDB Partition Selection exit routine.

Table 30. HALDB partition selection exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	The name given to the load module used for partition selection appears in the DBD associated with the database, the HALDB Partition Definition utility, or the DBRC INIT.DB command. The load module name must be the value of the parameter of the PSNAME operand on the DBD statement, Partition Selection name in the HALDB Partition Definition utility, or value of the parameter PARTSEL on the DBRC INIT.DB command.
Binding	After you compile and test your exit routine, bind it into IMS.SDFSRESL, SYS1.LINKLIB, or any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control and SAS regions.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSMPL.

Loading and deleting the routine

One HALDB Parttion Selection exit routine can be shared by multiple HALDBs. A HALDB Partition Selection exit routine can be placed in the IMS.SDFSRESL, SYS1.LINKLIB, or any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control region and SAS region.

When a HALDB definition in the RECON data set includes a HALDB Partition Selection exit routine definition, IMS loads the exit during IMS initialization if the HALDB is resident, during the first application scheduling if the HALDB is non-resident, or at the /START DB *partition_name* OPEN or UPDATE DB NAME(*partition_name*) START(ACCESS) OPTION(OPEN) command if the exit has not already been loaded.

When a HALDB database is taken offline, the associated HALDB Partition Selection exit routine is logically deleted from system memory. When all HALDB databases sharing a HALDB Partition Selection exit routine are offline, the exit routine is physically deleted from system memory. The following commands will delete the exit routine:

- UPDATE DB NAME(*HALDB_master_name*) STOP(ACCESS)
- UPDATE DB NAME(*HALDB_master_name*) STOP(UPDATES)
- /DBR DB *HALDB_master_name*
- /DBD DB *HALDB_master_name*

When a HALDB Partition Selection exit routine is not loaded, you can update or refresh the exit routine in the library where it is stored.

Calling the routine

IMS loads this routine at IMS initialization time.

The HALDB Partition Selection exit routine receives control during modification of the internal partition definition control block and when a DL/I call requires the selection of a partition. The following processing activities activate the HALDB Partition Selection exit routine:

- Control block initialization
- Control block termination
- Control block modification
- Selection of first partition
- Selection of next partition
- Selection of target partition

IMS calls a HALDB Partition Selection exit routine when an exit routine is specified for the database. When the internal partition definition control blocks are created, modified, or terminated, this call to the exit routine allows your exit to be aware of the current configuration of the HALDB partitions and to have some influence on its validity for subsequent DL/I processing. The initialization call that indicates that the control blocks were created occurs prior to authorizing and opening the partition data sets.

Cross memory mode

The following factors determine whether your HALDB Partition Selection exit routine is called in cross-memory mode:

- The IMS environment, either online (DLI) or batch (DBB)
- The call type, either control block manipulation or partition selection

Call type	Cross memory mode in batch environment	Cross memory mode in online environment
Control block manipulation calls	No	No
Partition selection calls	No	Yes

Communicating with IMS

IMS communicates with the HALDB Partition Selection exit routine through the entry registers.

Contents of registers on entry

The HALDB Partition Selection exit routine is called with the following registers established:

Register
Contents

1
Specifies the address of the parameter list that identifies the call. The parameters are:

1
A full word that contains the number of parameters in the list. The value of 2 is specified.

2
The Exit Communication Area that is mapped by DFSPECA.

3
The Partition Definition Area that is mapped by DFSPDA.

13
Address of a standard save area. Four pre-chained save areas are provided for this exit routine to use.

14
Return address to IMS.

15
Exit entry point address.

Area mapping

DFSPECA
Partition Exit Communication Area Mapping. Dynamically initialized from static storage.

DFSPDA
Partition Definition Area Mapping. Allocated and initialized during internal partition definition control block initialization.

Contents of registers on exit

The HALDB Partition Selection exit routine is involved in the processing of internal partition definition control block initialization, termination, rebuild, and partition selection. The exit routine can identify some processing and control block conditions as errors, according to your specifications. The exit routine informs IMS of the response to the error condition by specifying a return code. The return code is returned in field PECRC of the Partition Exit Communication Area (DFSPECA). The action that IMS takes depends on both the return code that is supplied by the exit routine and the call reason for invoking the exit routine. The exit routine can request IMS to stop the database or issue a pseudo abend. The following return codes can be sent to IMS:

Return code
Description

0
Normal return. No exception processing required.

4
Abnormal return. IMS can stop the database during control block calls, and IMS passes a status code FM back to the application program.

8
Pseudo abend return. IMS issues user abend 3499.

12
Exception return. No more partitions are available for Select Next processing. IMS treats this condition as the end of the HALDB.

Depending on the call reason and call history, IMS takes certain actions when return code 12 is received from the HALDB Partition Selection exit routine. The rules are as follows:

1. When the exit routine is called for control block initialization, termination, or modification (rebuild), the return codes can be 0, 4, or 8. A return code of 12 or above is not supported. The return code from a control block termination (PECTERM) call is ignored by IMS if it is 0, 4, or 8 (12 and above are not supported). IMS terminates the control block in all cases when the return code is 0, 4, or 8 for the PECTERM call.
2. When the exit routine is called for partition selection, return codes 0, 4, 8, and 12 are supported. If the partition selection is "Select Next", return code 12 from the exit routine indicates that no partitions are available. If the partition selection is "Select Target" or "Select First", return code 12 indicates a request for ABEND 3499.
3. When the exit routine is called for any partition selection, a check is made to see whether any prior call from the control block initialization, termination, and rebuild has resulted in a pending request for ABEND 3499. If such a request has been made, ABEND 3499 is issued.

Related tasks
Creating HALDB databases with the HALDB Partition Definition utility (Database Administration)

Related reference
“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Database Description (DBD) Generation utility (System Utilities)

INIT.DB command (Commands)

Sample partition selection exit routine (DFSPSE00)

DFSPSE00 contains code that supports control block initialization, termination, and modification calls, as well as partition selection calls.

Be aware that the actual partition selection processing in the sample DFSPSE00 is based on a high key value and not a user defined string value. The sample exit is written in assembler language and located in the IMS Sample library.

The sample exit routine demonstrates the use of the interface and control blocks. The sample exit performs partition selection processing by using partition high key.

Partition exit communication area mapping (DFSPECA)

The HALDB Partition Selection exit routine uses the DFSPECA communication area to communicate the result of exit processing.

The DFSPECA storage area is dynamically initialized from static storage for each invocation of the HALDB Partition Selection exit routine. The DFSPECA DSECT can be obtained by assembling DFSPSEIB.

DFSPECA Field Definitions:

Field
Description

PECDBN
The name of the HALDB.

PECRSWD1
Not used; the contents are unpredictable.

PECRC
Return code indicating the result of exit processing.

PECFDB
Exit feedback area consisting of two halfword fields.

The exit returns the partition ID of the partition selected in field PECFDB2.

PECKEY
Address of the key associated with the DL/I call.

PECCPID
Current Partition ID.

The partition ID of the last partition selected.

PECKEYL
The length of the key minus 1.

PECACT
The invocation action informing the exit what processing is required.

PECFLAG1
IMS control data flag. Defines additional information for exit processing.

PEC1NEWP
A flag indicating that one or more new partitions were added to the internal partition definition control block. To indicate that the entry defines a partition that was not previously defined, set flag PDAFLAG1 to PDAF101 in each related PDA entry. Set the flag for the exit REBUILD call and then reset.

PECFLAG2
Flag byte available for exit use.

Note: The user exit can set PECFLAG2 to any value, but that value is not preserved across calls to the exit routine.

PECVRSN
A halfword with the value PECURVER that is set by IMS before invoking the partition selection exit. The user exit can check the version number in PECVRSN with the constant PECURVER to ensure it is using the same or higher version of the DFSPECA control block passed by IMS. If the PECVRSN value is less than PECURVER value, a mismatch exists because the exit has been compiled with a higher version of the DFSPECA than the one used by IMS.

PECUSER
Dynamic work area for exit use. This work area storage is not preserved across calls to the exit routine.

Partition definition area mapping (DFSPDA)

The HALDB Partition Selection exit routine uses the DFSPDA partition definition area to define internal partition control blocks.

The DFSPDA storage area is allocated and initialized during initialization of the internal partition definition control block. DFSPDA storage area is maintained until the control block changes. Any control block change causes the storage to be released and a new area allocated and initialized. Each invocation of the HALDB Partition Selection exit routine passes the DFSPDA area. The DFSPDA DSECT can be obtained by assembling DFSPSEIB.

DFSPDA field definitions

PDAPDE
The address of the first partition definition entry.

PDANUM
The number of DFSPDA entries.

PDARSWD1
Not used; the contents are unpredictable.

PDALSTRL
The length of the longest string that is defined for the partitions.

PDADORG
The database organization: PHDAM, PHIDAM, or PSINDEX.

PDAUSRn
Five words that are available for exit use (PDAUSR1, PDAUSR2, PDAUSR3, PDAUSR4, and PDAUSR5).

The exit routine can use these words to anchor storage that has been allocated by the exit and these values will be available for use the next time the exit is called. The exit can also use the GETMAIN and FREEMAIN macros.

PDAPLEN
The length of the Partition Definition Area Prefix.

DFSPDAE field definitions

PDAPN
The name of the associated partition.

PDASTRG
The address of the user-defined Partition String value. If PDASTRG is zero, it indicates a null Partition String. This 256–byte area contains the string value that you define. You can modify this area during Structure Initialization processing to assist in selection processing.

PDAPID
The partition ID of the associated partition.

PDARAP
The number of Root Anchor Points defined for the partition. Provided for PHDAM organization only; otherwise, it contains zeros.

PDABLKR
The number of blocks containing Root Anchor Points. Provided for PHDAM organization only; otherwise, it contains zeros.

PDASTRGL
The length of the user string minus 1.

PDAFLAG1
IMS control data flag. Defines unique PDA entry information for exit processing.

PDAF101
A flag within PDAF101 indicating whether this PDA entry defines a new partition that was not previously defined. When PDAF101 is on for the control block modification call, it indicates that this entry is for a new partition; when off, PDAF101 indicates a previously defined partition.

PDAELEN
The length of the Partition Definition Area entry.

Length added to the entry address to provide the address of the next entry.

HDAM and PHDAM randomizing routines (DFSHDC40)

The DL/I HDAM and PHDAM access method requires you to supply a randomizing module for placing root segments in, or retrieving them from, an HDAM and PHDAM database.

This topic contains Product-sensitive Programming Interface information.

Subsections:

- "About these routines" on page 104
- "Communicating with IMS" on page 106
- "Sample HDAM and PHDAM randomizing routines" on page 107

About these routines

Several databases can share the same routine, but each of those databases must be associated with a single randomizing routine. If you are using data sharing, you must use the same randomizing routine on all systems that share a given database.

A randomizing module uses a mathematical technique to convert a key into an address. A specific key always converts to the same address. The randomizing module required by IMS must convert a key field value into a relative block number and an anchor point number. The result of a randomizing routine is a relative block number that ranges from 1 to 2^{24}-1. The anchor point number ranges from 1 to the number of anchor points per block as defined in the database's DBD. The maximum is 255.

The key field value is supplied by an application program in the data itself for inserting segments into the database and in an application program in an SSA (segment search argument) for retrieving segments from a database.

Four randomizing modules are supplied with IMS. Although four are supplied, DFSHDC40 is the only one recommended for use. You can use this one or write your own randomizing module.

Related Reading: To help you determine the module that best meets your need, see *IMS Version 15.2 Database Administration*.

If you write your own module, follow the guidelines included in this topic.

Attributes of the routine

The following table shows the attributes of the HDAM and PHDAM Randomizing routine.

Table 31. HDAM and PHDAM randomizing routine attributes

Attribute	Description
IMS environments	DB/DC and DBCTL.

Table 31. HDAM and PHDAM randomizing routine attributes (continued)

Attribute	Description
Naming convention	The name you give to the load module used for randomizing functions with a specific database must appear in the DBD generation associated with the database. The load module name must be the value of the "mod" parameter of the RMNAME= operand on the DBD statement in the HDAM and PHDAM DBD generation. **Related Reading:** For details on coding this parameter, see "Database description (DBD) generation", in *IMS Version 15.2 System Utilities*.
Link editing	After you compile and test a randomizing module, bind it into IMS.SDFSRESL, SYS1.LINKLIB, or into any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control, SAS, and batch regions. To ensure that the routines run as they did in prior IMS releases, bind them as neither reentrant nor reusable.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	For the latest version of the sample routine (DFSHDC40), see IMS.ADFSSRC; member name is DFSHDC40.

You must write, compile, and bind the randomizing module as one of the following:

REENTRANT
IMS does not serialize the database before calling the routine. A single copy of the routine is used for the databases.

REUSABLE
IMS serializes the database before calling the routine. If the routine is used for multiple databases, it must be written and compiled as reentrant, even if it is not bound as reentrant.

NONREUSE
IMS serializes the database before calling the routine. Each database has its own copy of the routine.

All modules receive control and must return control in 31-bit addressing mode. They must be able to execute in cross-memory and task modes.

Loading the routine

IMS loads all randomizing modules from their resident library when the database is opened. IMS obtains the name of the randomizing module from the name you have specified in the RMNAME parameter of the DBD statement of the database description (DBD).

Related Reading: For details on coding the RMNAME parameter, see *IMS Version 15.2 Database Utilities*.

The necessary randomizing module associated with a specific database is brought into main storage at the time the associated database is opened. It can also be placed in the LPA (link pack area). This allows one copy of the module to service several databases that are concurrently open.

If you use any of the Local Storage Options (LSO), the randomizing module is loaded in CTL or DL/I SAS private storage. Otherwise, the module is loaded into CSA.

Calling the routine

When an application program issues a Get Unique or Insert call that operates on a root segment of an HDAM and PHDAM database, the randomizing module is called.

The source of the root key that IMS supplies to the randomizing routine is as follows:

- For a root insert, IMS takes the key from the I/O area containing the root to be inserted.
- For a call qualified on the root key, IMS uses the key value in the segment search argument.

Related Reading: For information on processing Get Next (GN) calls qualified on the root key and calls with root qualification that allows a range of key values, see *IMS Version 15.2 Application Programming*.

The key is supplied to the randomizing module for conversion to a relative block number and anchor point number within the database. In addition to the key supplied by an application program, parameters from the DBD generation for the database are available to the randomizing module.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the randomizing routine.

Contents of registers on entry

On entry, the randomizing routine must save all registers using the provided save area. The registers contain the following:

Register	Content
0	Address of Data Management Block (DMB).
1	Address of the DMBDACS CSECT.
7	Address of Partition Specification Table (PST).
9	Address of first byte of key field value supplied by an application program.
13	Address of save area. The exit routine must not change the first three words.
14	Return to IMS address.
15	Entry point of randomizing module.

If an HDAM and PHDAM database does not have a sequence field defined:

- The executable key length field in the CSECT named RDMVTAB is not initialized and must not be used.
- The value in register 9 module contains the address of the first byte of the segment.

If an HDAM and PHDAM database does not have a sequence field defined at the root level, the randomizing module is given control on an insert call. All retrieval calls result in a scan of the root-level qualification. On Get Unique (GU) calls, the scan starts at the beginning of the database. On Get Next (GN) calls, the scan starts at the current root-level position within the database.

The randomizing module is invoked on Get calls, particularly when the database contains a secondary index or a logical relationship. The randomizing module must produce the same results on the Get call as it did on the Insert call.

The first eight words of the PST are available to the randomizing module as a work area. These words are also used by DL/I and must not be used by other exit routines. If an additional work area is needed, CSECT RDMVTAB can be expanded to provide additional space.

Internal IMS control blocks that can be of value to a randomizing routine are the Partition Specification Table (PST), the Physical Segment Description Block (PSDB) for the root segment, and the first Field Description Block (FDB). The FDB is the root segment key field format description.

Description of parameters

The parameters from DBD generation are available to randomizing modules. Their area is described by the DMBDACS DSECT. It contains information such as the randomizing routine's name, anchor point information, and the total area length. You can extend the area by an assembly and bind process to contain any data or algorithm information.

The root 32 bytes of the RDMVTAB CSECT (described by the DMBDACS DSECT) contains constants defined by DBDGEN. If you extend the area to include additional parameters, this field must be

duplicated. The DMBDASZE field must be updated to reflect the total length of this area (including the added parameters).

After assembly, you can bind the expanded RDMVTAB CSECT to replace the old one. Use an ENTRY statement specifying the name of the DBD and an ORDER statement to make sure the original order of the multiple CSECTs is maintained. For more information, see information on the z/OS binder and loader in the z/OS product library.

The following DSECT defines the format of the area pointed to by register 1:

```
DMBDACS   DSECT
DMBDANME  DS     CL8   NAME OF ADDR ALGORITHM LOAD MODULE
DMBDAKL   DS     CL1   EXECUTABLE KEY LENGTH OF ROOT
          DS     CL3
DMBDASZE  DS     H     SIZE OF THIS CSECT
DMBDARAP  DS     H     NUMBER OF ROOT ANCHOR POINTS/BLOCK
DMBDABLK  DS     F     NUM OF HIGHEST BLOCK DIRECTLY ADDRSD
DMBDABYM  DS     F     MAX NUMBER OF BYTES BEFORE OFLOW TO
                       2NDARY
DMBDARC   DS     CL1   RETURN CODE FROM RANDOMIZER
          DS     CL3   RESERVED
DMBDACP   DS     F     RESULT OF LAST ADDRESS CONVERSION
```

Contents of registers on exit

Before returning to IMS, the randomizing routine must restore all registers. The parameter list pointed to by register 1 can contain one of the following return codes:

Return code	Meaning
0	Continue processing; randomizing properly.
4	Set FM status code and return to caller.
8	U812 abend.

For any randomizing routine that passes these return codes, ensure that application programs that use the database can accept the return codes.

The return code from a randomizing module can be in either character or binary form. In other words, X'F0' and X'0' are both valid for a return code of zero. This return code must be placed in the DMBDARC field of the CSECT addressed by register 1.

You do not need to explicitly set a return code of zero in DMBDARC, because it is the default return code and the field is preset to zero.

Results of the routine on exit

The result of a randomizing module conversion must be in the form *BBBR* where BBB is a 3-byte binary number of the block into which a root segment is inserted or from which it is retrieved and R is a 1-byte binary number of the appropriate anchor point, within a relative block, within a data set of the database.

This result must be placed in the CSECT addressed by register 1 in the 4-byte fixed name DMBDACP. If the result exceeds the content of the field DMBDABLK, the result is changed to the highest block and last anchor point of that block.

Sample HDAM and PHDAM randomizing routines

IMS supplies four randomizing module samples (DFSHDC10, DFSHDC20, DFSHDC30, and DFSHDC40) to help you write your own HDAM and PHDAM randomizing module. The modules are linked into the IMS.SDFSRESL data set during system definition. The modules use the following randomizing techniques:

- Modular or division method (DFSHDC10)
- Binary halving method (DFSHDC20)
- Hashing method (DFSHDC30 and DFSHDC40)

Module DFSHDC40 is recommended; the source code for all four modules resides in the IMS.ADFSSRC library. The next provides guidelines for using the sample module, DFSHDC40.

Restriction: These routines do not support nonsequenced HDAM and PHDAM databases. They all use the key length in their calculations.

Related concepts

"Guidelines for writing IMS exit routines" on page 3
Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

Sample HDAM and PHDAM generalized randomizing routine (DFSHDC40)

You can use the IMS-supplied sample HDAM randomizing modules DFSHDC40 on IMS.ADFSSRC.

If root keys are unique and totally random storage is desired, this routine can be used for any HDAM and PHDAM database without performing an analysis of key distributions.

This randomizing routine works with the entire key and has the following characteristics:

- It is reentrant.
- Keys can contain any of the 256 characters, and key length can be from 1 to 256 bytes.
- It converts any key distribution (with unique key values) to a totally random address distribution.
- It never returns an address in block 1, which is always a bit map block in HDAM and PHDAM. You can specify any number of blocks and RAPs.
- The number of blocks must be in the range between 2 and 2^{24}-1; the number of RAPs must be in the range of 2 to 2^{31}-1 when RAPs are multiplied by blocks. The RBN subparameter of the RMNAME= parameter of the DBD statement must be specified for the upper limit, together with DFSHDC40 as the "mod" subparameter, if this randomizing routine is chosen.
- It allows the insertion of a dummy root at the highest block-RAP to ensure the formatting of the entire root addressable area at load time.

The basic logic of the routine is:

1. Convert the key into a 4-byte binary number by translating the key digits twice. Determine the offset into the translation table using the key length and individual digits. For example:

   ```
   Key      123456
   ```

   Digits are used in series of threes. Two work areas are used. In the first pass, the first work area contains X'F2F3'; the second contains X'F1F2F3'.

   The first work area is translated into the translation table with a zero point of 4 (key length 2). The second work area is translated into the translation table with a zero point of X'F5', the fifth digit. These two translated numbers are multiplied and added into an accumulator. The remaining digits are converted and added into the accumulator.

   The conversion number for key 123456 is X'45683199'.
2. Translate the converted number, and set the top bit to zero to ensure a positive number.
3. Multiply the maximum number of blocks minus one by the number of RAPs. Multiply the result by the translated key.
4. After adjustment to ensure block 1 is not used, store the result in DMBDACP.

Secondary Index Database Maintenance exit routine

You can use the Secondary Index Database Maintenance exit routine to control the density of a secondary index by selectively suppress secondary indexing.

Subsections:

- "About this routine" on page 109

- "Communicating with IMS" on page 110

About this routine

Two options are available to the Database Manager to control the volume of entries in secondary index databases: the NULLVAL operand and the index maintenance exit routine. To build and maintain a sparse index, you can use suppression of indexing, the process of withholding a prospective index pointer segment from the index.

Use the NULLVAL operand to suppress indexing when the entire indexed field contains one specified character or value. For example, you might want to use NULLVAL to suppress indexing when the indexed field contains only blanks. A different NULLVAL can be specified for each indexed segment.

Alternatively, secondary indexing allows you to specify, during the DBDGEN, a user-supplied exit routine that can selectively suppress secondary indexing. One exit routine is allowed for every secondary index; however, one generalized routine can be written to serve several index relationships.

If you bind this exit routine as reentrant (RENT), it must be truly reentrant (it cannot depend on any information from a previous invocation and it cannot store into itself).

If you bind this exit routine as reusable (REUSE), it must be truly reusable (it cannot depend on any information in itself from a previous call), but it can depend on information that it saves in the specific database segment block that is passed to it. In addition, if the same exit routine is used for two different segments, the single copy of the exit can be called concurrently for each segment. In this case, the exit routine must be written as reentrant.

If you bind this exit routine so that it is neither RENT nor REUSE, it can store into itself and depend on the information saved in the database segment block that is passed to it.

The following table shows the attributes of the Secondary Index Database Maintenance exit routine.

Table 32. Secondary index database maintenance exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	Each exit routine must have a name unique with respect to all IMS module names and to any other exit routines in the IMS libraries. The name of this exit routine is specified for each DBD with the EXTRTN parameter of the XDFLD statement submitted to the DBDGEN utility. Before an index source segment in a database can be loaded or updated, its EXTRTN routine must be in the system library.
Link editing	After an exit routine has been compiled and tested, it can be placed into the IMS.SDFSRESL data set, from which it is loaded by IMS. It can also be placed in SYS1.LINKLIB, or in any operating system partitioned data set to which access is provided with a JOBLIB or STEPLIB JCL statement.
Including the routine	No special steps are need to include this routine.
IMS callable services	This exit routine is not eligible to use IMS callable services.

Loading the routine

The first time that an exit routine associated with the specific database is referenced, it is loaded into storage in either the IMS online control program region or batch processing region when the associated database is opened. The loaded routine will be used by any other databases that require the same exit routine. This allows one copy of the module to service several databases that are open concurrently. The routine is not refreshed during the current IMS execution.

When an index maintenance exit routine is used in either the IMS online control region or a DL/I batch processing region and the exit routine does not exist in LINKPACK, you must provide space in the IMS

control region or in the DL/I separate address space (DLISAS) to accommodate the exit routines that can be used for online databases.

Calling the routine

When an application program issues a REPL, ISRT, or DLET call of a segment serving as an index source segment for one or more indexing relationships, the DL/I index maintenance routine is invoked.

DLET call

In the case of DLET, an indexing segment is built corresponding to the existing index source segment. If it passes the null value test, the index exit routine is invoked. This routine indicates whether this indexing segment should appear in the index. If it should appear, the actual indexing segment is retrieved and deleted; otherwise, no delete is attempted.

ISRT call

In the case of ISRT, the indexing segment is built to correspond to the segment to be inserted, and the null value test and the exit routine tests are performed. If no suppression of indexing is indicated by either, it is inserted into the index.

REPL call

A REPL call can be a combination of a DLET call and an ISRT call, a simple replace, or a NOP, depending on the fields changed in the replace. If a field in the Index Source Segment (ISS) is changed by a REPL call that changes the indexed data or subsequent data, the existing indexing segment is deleted and a new one inserted. The index edit routine is invoked for each operation. If the change in the ISS affects a source data field, a replace operation on the indexing segment is executed, unless the index exit routine indicated that indexing was suppressed. If the ISS replace made no changes in the indexing segment, no action is taken.

The suppression of indexing by the exit routine must be consistent. The same indexing segment cannot be examined at two different times and have suppression indicated only once. If the indexing segment contains user data, this user data cannot be used to evaluate suppression, since the actual indexing segment is seen by the exit routine just before the insertion of a new one. In the cases of replace and delete, only a prototype is passed. The prototype contains the constant, indexed data, subsequence data, duplicate data, and any symbolic pointer that was added. Therefore, index suppression must not be based on any user data.

The exit routine issues a return code and indicates either that the present index pointer segment belongs in the index or that it should be suppressed. The exit routine must not change any IMS control blocks, or any fields in the indexing segment.

You can include additional information about the segment in the exit routine CSECT. This CSECT is part of the DBD, and as such can be replaced by a bind. It is of variable-length and contains a fixed-format header. A separate CSECT is provided for each XDFLD in the DBD for which an exit routine is specified. The availability of this CSECT is described in the exit routine specifications. You can replace this control section in the same manner as you can the segment compression control section.

Communicating with IMS

IMS communicates with the exit routine through the entry and exit registers.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of Partition Specification Table (PST).
2	Address of proposed or existing index segment.
3	Address of Index Maintenance Routine Parameters CSECT.

Register	Contents
4	Address of Index Source Segment.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Description of parameters

On entry to the routine, IMS passes the address of the CSECT shown in the following figure.

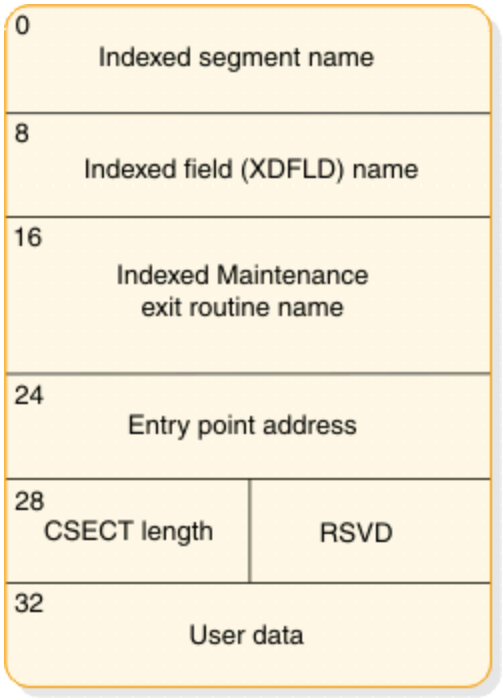


Figure 5. Index maintenance exit routine parameter list CSECT

The following DSECT defines the format of this CSECT:

```
DMBXMPRM    DSECT
DMBXMSGN    DS      CL8     Name of indexed segment
DMBXMXDN    DS      CL8     Name of indexed field
DMBXMXNM    DS      CL8     Name of exit routine
DMBXMXEP    DS      A       Entry point addr
DMBXMPLN    DS      H       Total length of CSECT
            DS      H       Not Used
```

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except register 15, which contains one of the following return codes:

Return code	Meaning
0	The indexing segment should appear in the index for this database segment.
4	Indexing should be suppressed.

Related reference

"Routine binding restrictions" on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample Secondary Index Database Maintenance exit routine

The sample secondary index database maintenance exit routine shows entry and exit code to help you write your own routine.

The following secondary index database maintenance exit routine example is not a usable exit routine provided by IMS, nor is it found in the IMS.SDFSSMPL library.

```
SAMPLE   TITLE 'SAMPLE OF SECONDARY INDEX EXIT ROUTINE'
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
* SAMPLE OF SECONDARY INDEX DATA BASE MAINTENANCE EXIT ROUTINE          *
*                                                                       *
* THIS SAMPLE IS NOT INTENDED TO BE A USABLE EXIT ROUTINE.              *
* IT IS PROVIDED HERE TO SHOW ENTRY AND EXIT CODE.                      *
* THIS SAMPLE SUPPRESSES THE INDEX ENTRY IF ALL BYTES OF THE            *
* INDEX KEY ARE BLANK.                                                  *
*                                                                       *
*                                                                       *
*    REGISTERS ON ENTRY                                                 *
*      R1  - PARTITION SPECIFICATION TABLE (PST) ADDRESS                *
*      R2  - ADDRESS OF (PROPOSED OR EXISTING) INDEX SEGMENT            *
*      R3  - ADDRESS OF INDEX MAINTENANCE ROUTINE PARMS CSECT           *
*      R4  - ADDRESS OF INDEX SOURCE SEGMENT                            *
*      R13 - SAVE AREA ADDRESS                                          *
*      R14 - RETURN ADDRESS                                             *
*      R15 - ENTRY ADDRESS                                              *
*                                                                       *
*    REGISTERS ON EXIT                                                  *
*      R15 - 0 TO NOT SUPPRESS THE INDEX ENTRY                          *
*          - 4 TO SUPPRESS THE INDEX ENTRY                              *
*      R0 THRU R13 ARE RESTORED                                         *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
INDEXXIT CSECT
         STM   R14,R12,12(R13) SAVE REGISTERS 14 THRU 12
         L     R13,8(R13)      SET 13 TO NEXT IMS PRE-CHAINED SAVE SET
         LR    R12,R15         SET 12 AS BASE
         USING INDEXXIT,R12    USE R12 AS BASE FOR PROGRAM
         USING PST,R1          USE R1 AS BASE FOR PST
         USING XRECORD,R2      USE R2 AS BASE FOR INDEX RECORD
         USING DMBXMPRM,R3     USE R3 AS BASE FOR INDEX CSECT
         USING XSOURCE,R4      USE R4 AS BASE FOR INDEX SOURCE SEGMENT
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
*  LOGIC SHOULD BE PROVIDED HERE TO DECIDE WHETHER THE INDEX RECORD     *
*  SHOULD BE SUPPRESSED.                                                *
*                                                                       *
*  THE FOLLOWING CODE WILL TEST WHETHER THE KEY OF THE INDEX            *
*  RECORD IS ALL BLANK.  IF THE FIELD IS ALL BLANK, THE INDEX ENTRY     *
*  WILL BE SUPPRESSED.                                                  *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
         CLC   XFIELD1,BLANKS  IS FIELD BLANK
         BE    SUPPRESS        YES, SUPPRESS INDEX FOR FIELD
         B     NOSUPP          NO, ALLOW INDEX FOR FIELD
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
* SUPPRESS RETURN, SET 4 IN R15 TO TELL IMS TO SUPPRESS THE ENTRY       *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
SUPPRESS DS    0H
         L     R13,4(R13)      BACK UP TO PRIOR SAVE AREA
         RETURN (14,12),RC=4   RETURN WITH 4 IN R15
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
* NORMAL RETURN, SET 0 IN R15 TO TELL IMS TO NOT SUPPRESS THE INDEX     *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
         SPACE 1
NOSUPP   DS    0H
         L     R13,4(R13)      BACK UP TO PRIOR SAVE AREA
         RETURN (14,12),RC=0   RETURN WITH 0 IN R15
         SPACE 2
BLANKS   DC    CL255' '        CONSTANT OF 255 BLANKS
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE DSECT FOR THE INDEX RECORD                                 *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
XRECORD  DSECT
XFIELD1  DS    CL5
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE DSECT FOR THE INDEX SOURCE SEGMENT                         *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
XSOURCE  DSECT                 DSECT FOR INDEX SOURCE SEGMENT
XSFIELD1 DS    CL5             FIELD 1 OF INDEX SOURCE SEGMENT
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* DSECT FOR INDEX MAINTENANCE EXIT ROUTINE PARAMETER CSECT            *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
DMBXMPRM DSECT
DMBXMSGN DS    CL8             NAME OF INDEXED SEGMENT
DMBXMXDN DS    CL8             NAME OF INDEXED FIELD
DMBXMXNM DS    CL8             NAME OF USER EXIT ROUTINE
DMBXMXEP DS    A               EXIT ROUTINE ENTRY POINT ADDRESS
DMBXMPLN DS    H               TOTAL LENGTH OF CSECT
         DS    H               NOT USED
DMBUSERD DS    C               START OF USER DATA IF ANY
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE DSECT FOR THE IMS PST WHICH IS PASSED IN R1                *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
         PRINT NOGEN
         IDLI  PSTBASE=0
         PRINT GEN
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE EQUATES FOR SYMBOLIC REGISTERS                             *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
         REQUATE
         SPACE 2
         END
```

Segment edit/compression exit routines

You can write a segment edit/compression exit routine to compress and expand segments of data.

This topic describes the segment edit/compression exit routine, its attributes, how to activate it, how the routine communicates with IMS, and the restrictions that apply. The topic also provides a description of sample segment compression/expansion modules.

Subsections:

- "About this routine" on page 114
- "Restrictions" on page 119
- "Communicating with IMS" on page 120

About this routine

Segment compression saves space and can result in reduced logging. You can write an exit routine to:

- Edit or compress both fixed- and variable-length segments
- Accomplish either data edit/compression (DEDBs or full-function databases) or key edit/compression (full-function databases only).

If you write your own exit routine, you can also allow for editing, such as encoding and decoding segments for security purposes, and for validating and formatting data. The logic for data encoding and decoding (or for other desired editing or formatting) can be based on information contained within the user-written routine itself. It also can be based on information from an external source, such as data provided in the DBD block, or from tables examined at execution time.

Segment compression is possible for both full-function databases and data entry databases (DEDBs). You can use either DFSCMPX0 or DFSKMPX0, write your own, or generate one which invokes hardware data compression.

You can apply the same exit routine to multiple segment types within the same or different databases.

Recommendation: Use the DFSCMPX0 sample routine, because it uses z/OS services.

The segment edit/compression exit routine is optional. No default routine is called. The sample exit routines only perform segment compression and expansion. The exit routines should be implemented by those having overall systems or database responsibility for an installation. These routines should be transparent to the application programs that access the databases.

Related Reading: For a list of the specific full-function databases that are supported and for additional guidance-level information, see *IMS Version 15.2 Database Administration*.

Restriction: The DEDB Sequential Dependent Scan utility (DBFUMSC0) provides support for SDEP segment decompression only if the EXPANDSEG command is specified.

Related Reading: For details on coding the EXPANDSEG command, see *IMS Version 15.2 Database Utilities*.

The following table shows the attributes of the segment edit/compression exit routine.

Table 33. Segment edit/compression exit routine attributes

Attribute	Description
IMS environments	All environments that support databases.
Naming convention	According to user's naming convention.
Link editing	After an edit routine has been compiled and tested and before it is used by the IMS system, it must be placed into IMS.SDFSRESL, SYS1.LINKLIB, or into any operating system partitioned data set to which access is provided with a JOBLIB or STEPLIB control region JCL statement. You must also specify one entry point to the exit routine.
Including the routine	Routine is specified in the SEGM macro for the DBDGEN.
IMS callable services	To use IMS callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the PST address found in register 1 as the ECB. • Link DFSCSI00 with your user exit.

Table 33. Segment edit/compression exit routine attributes (continued)

Attribute	Description
Sample routine location	IMS.ADFSSMPL.

Attributes of the Routine

The following list describes the attributes of the segment edit/compression exit routine.

Minimum Authorization
Supervisor state in key 7.

APF Authorization
Must reside in either in IMS.SDFSRESL, SYS1.LINKLIB, or in an authorized PDS library specified in JOBLIB or STEPLIB. It can also reside in any library specified in LNKLSTxx of SYS1.PARMLIB. It can be in SYS1.LPALIB only if the library is included in IEAAPFxx of SYS1.PARMLIB.

Cross Memory Mode
Exit can be entered in cross-memory mode in the online environment but not in batch mode.

AMODE, RMODE
Exit resides in 24-bit and can be entered only in 24-bit.

Handling Abnormal Conditions
Any error conditions that are returned by system services on compression/expansion are handled by the sample routine DFSCMPX0, which sets register 0 and register 15 with abend code 2990 and reason code before returning to caller. See the reason codes in Table 26. However, the action modules normally pseudoabend the application with a U840 abend.

The following attributes of the segment edit/compression exit routine differ depending on the type of database that uses the routine.

Full-Function Database
The exit routine must be coded to be serially reusable.

IMS does not reload the routine between consecutive calls to the exit. IMS loads the routine once per segment reference. If the exit is link-edited as reusable (REUS), the same physical copy of the load module in storage is used to satisfy all load requests.

Because IMS calls the exit by branch and link, there is no operating system serialization of exit calls. IMS internally serializes calls to full-function database compression exits at the database level. For HALDB database compression exits, this means that all calls to the database, regardless of which partition the segments reside in, are serialized through the compression exit.

If the same exit name is used across more than one database or is used in a HALDB database organization, the exit must either be coded and link-edited (bound) as reentrant and reusable, or it must be coded as reusable but link-edited as not-reusable. If the exit is link-edited as not-reusable, a separate copy of the exit is loaded for each segment reference and used only by that segment reference. Code the exit as logically reentrant so that it is also serially reusable.

DEDB
If the segment edit/compression exit routine is used with DEDBs, you must write it and bind it as reentrant. In addition, the exit routine is loaded during control region initialization rather than during the opening of a database (as it is with a full-function database).

Loading the routine

Each time a database is opened, IMS examines each segment description to determine whether edit/compression has been specified for that segment type. If so, the exit routine is loaded from its resident library by IMS. IMS obtains the name of the routine from the COMPRTN parameter of the SEGM statement of the DBD.

An IMS restart is required to refresh the loaded exit routine with a new version.

Related Reading: For details on coding the COMPRTN parameter, see *IMS Version 15.2 System Utilities*. Adequate storage for the edit/compression routine must be provided for both batch and online systems.

How the segment edit/compression facility works

When a segment requiring editing or compression is accessed, IMS gives your edit routine control and provides it with the following information:

- Address of the data portion of the segment.
- Address of the segment work area.

Definition: Although the exit can be used for functions other than compression, from this point on the use of the term *compression* refers to the process of converting the segment from the application program form to the form written to external storage. The term *expansion* refers to the process of converting the segment from the external storage form to the application program form.

Two types of segments can be presented to the routine: fixed length segments, with a data length that is static and is reflected in control blocks; and variable-length segments, with its data length contained within a field in the first two bytes of the segment itself. While a routine dealing with a single-segment type normally does not need to recognize the differences, a more general purpose module involved with multiple segment types can obtain sufficient information to differentiate between them. This is done by examining data provided in the segment compression control section.

Segments being processed using the segment edit/compression facility are stored as variable-length segments in the database. Variable-length segments have a size field in the first two bytes of the data portion of the segment. This size field defines the length of the data portion of the segment. When segments are defined to the application program as fixed length, your routine must expand it to the fixed length expected by the application program. In reverse, if the application program presents a fixed-length segment, your edit routine must add the size field to the compression segment. If the segment is a variable-length segment, it must update the size field with the correct segment length.

Example

Although your edit routine can modify the key fields in a segment, the segment's position in the database is determined by the original key field.

Example: If the key field of a segment type is based on last names and the database has segments for people named McIvor, Hurd, and Caldwell, these segments are maintained in alphabetic sequence—Caldwell, Hurd, and McIvor. Assume your edit routine encodes the names as follows:

```
Caldwell ------> 29665
Hurd     ------> 16552
McIvor   ------> 24938
```

The encoded value is put in the key field. However, the segments in the database remain in their original sequence (Caldwell, Hurd, McIvor) rather than in the numeric sequence of the encoded values (16552, 24938, 29665). Because segments in the database are maintained in their original sequence, application programs can issue GN calls and retrieve the correct segment even though segments are encoded. This is also true for secondary index fields contained in index source segments.

Using the DBD table

The DBD control block has a table appended to it in the form of an assembler language CSECT. One CSECT is filled in for each segment type that specifies the use of the segment edit/compression facility. The CSECT contains basic information, such as the name of your edit routine and the name of the segment type. You can extend the CSECT to contain any editing parameters or criteria you want. In other words, some or all of the logic for editing a segment type can be put in the CSECT. You can perform different editing operations on different segment types with a single edit routine. If you want additional information for editing a segment type, any external source can provide it, not just the table in the DBD.

Related Reading: For information on the DBD control statement SEGM, see the section "SEGM Statement" in *IMS Version 15.2 Database Administration*.

Activating the routine

When the application program is activated and begins accessing segments, IMS interfaces with the segment edit/compression exit routine as described in this section. In all cases, IMS passes an entry code to the exit routine. Your exit routine must examine this entry code to determine the function to be performed.

Activating the routine for compression

For compression, regardless of the format at the source address, the segment at the destination address must be in variable-length format. The following figure shows the input (a fixed- or variable- length segment) in expanded format that is passed to the edit/compression routine and output (as a variable-length segment) in compressed format. The first data field of the destination segment is a 2-byte segment size field.

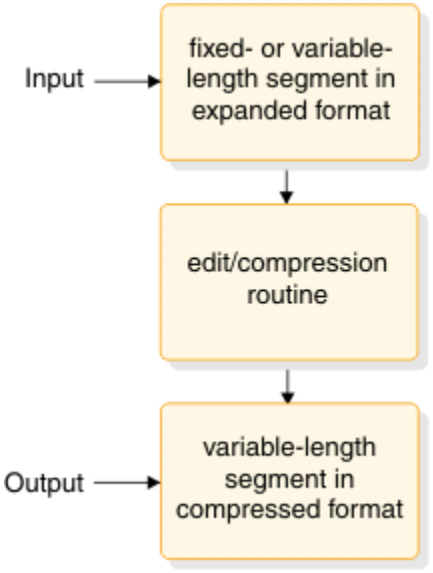


Figure 6. Segment compression

Segment length

If a fixed- or variable-length segment requires compression, and the data format is such that compression cannot take place, the addition of control information by your exit routine (indicating the segment could not be compressed) lengthens the segment beyond the maximum length definition. To allow for this expansion, and to allow IMS to check the validity of compression results, you can increase the size of your segment. You can increase the size of fixed-length segments by up to 10 bytes:

- For full-function fixed-length segments, you can increase the segment size by more than 10 bytes if the value for the COMPRTN parameter of the DBD SEGM statement specifies more. You can increase the size of a full-function variable length segment up to the maximum defined size.
- You can increase the size of a DEDB variable length segment up to the maximum defined size plus 10 bytes, but it must not exceed 120 bytes less than the control interval (CI) size.

The length of the segment to be moved is provided in one of two places:

- If the segment length specified in the DBD is fixed, the source length is in the DMBCPSGL field.
- If the segment is defined as variable in length, the source length is provided as a binary value in the first two bytes at the source address.

In either case, the move operation provided by the edit/compression routine must result in a 2-byte length field, followed by the corresponding quantity of data in the segment work area.

IMS might pad a segment to a length greater than that created by your exit routine. IMS pads full-function variable-length segments to their minimum length. IMS pads full-function fixed-length segments to their

pad length if it is specified on the COMPRTN parameter of the DBD SEGM statement. IMS does not pad DEDB segments.

Activating the routine for expansion

For expansion, the input segment has a variable-length format. The following figure shows the input (a variable-length segment) in compressed format that is passed to the edit/compression routine and output (as a fixed- or variable- length segment) in expanded format.

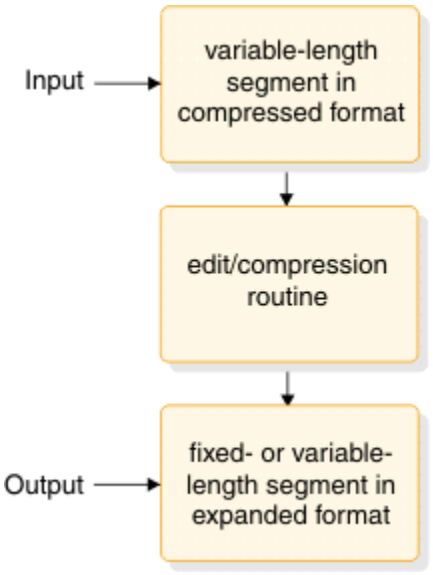


Figure 7. Segment expansion

Entry code determination

For segment expansion that occurs during the segment retrieval process, IMS examines the application program request. If the request is satisfied by a compressed segment, a test is made to determine the type of compression used, either key or data. Then, depending on the type of retrieval request, either entry code 4 or 8 is passed to the expansion routine. The following criteria are used as a basis for the decision:

- If the segment can be accepted without analysis of either a key or data field, control is transferred using entry code 4. The segment is expanded to the form presented to the user.
- If the value of the segment sequence field requires examination prior to segment selection, an additional check is performed to determine data or key compression. Data compression requires no additional processing, while key compression requires activation of entry code 8. If the segment is qualified for presentation after the key field is validated, IMS formats the segment using entry code 4 and passes it to the exit routine.
- If data field analysis is necessary to properly satisfy the DL/I call, proper expansion of the segment by entry code 4 occurs. When the correct segment is found, it is passed to the user.

The format of the segment presented through entry codes 4 and 8 of the compression routine is identical to that of a variable-length segment (a 2-byte segment size field followed by the appropriate quantity of data). The exit routine must expand the segment at the destination address in correct format, either fixed or variable-length. In the case of key compression, the exit routine must expand the segment from its start to the sequence field. For variable-length segments, the segment data length field, after processing by the key expansion, must reflect the length of the expanded portion of the segment at the destination address.

Using the routine with tabled data information

You have two options for processing tabled data information:

- Include the tabled data in the DBD module itself.

For each segment defined during DBDGEN as eligible for edit/compression, an entry is developed in an assembly language control section. This control section can be extended by assembling and binding it to contain any desired data or algorithm information.

- Load the tabled data when the exit routine is initialized.

  Specifying INIT on the COMPRTN parameter of the SEGM statement in the DBD causes the routine to be called for initialization processing. The routine can issue IMS callable services calls to provide functions equivalent to the LOAD/DELETE or GETMAIN/FREEMAIN macro instructions. These calls bring additional information into storage in the form of modules from IMS.SDFSRESL library. For example, the routine can maintain a table of substitution characters that is separate from the executable code. This table can reflect different combinations for different segments, resulting in a general purpose, table-driven routine capable of processing several segment types.

IMS provides two additional entry codes that allow you to process tabled data information. IMS calls a segment edit/compression exit routine with these entry codes if you specify the INIT keyword on the COMPRTN parameter of the SEGM statement. With these codes, IMS passes control to the initialization and termination subroutines immediately after the full-function database or DEDB area is opened, and immediately before the full-function database or DEDB area is closed. Any processing required for the database segments that cannot be directly related to any one segment can be done at this time using these options. Initialization processing and termination processing can include the loading and deleting of the compression algorithm table.

Code	Description
12	Initialization processing call. Control is obtained for algorithm initialization processing immediately after the full-function database or DEDB area is opened. Registers 2 and 3 are unpredictable.
16	Termination processing call. Control is obtained for algorithm termination processing immediately before the full-function database or DEDB area is closed. Registers 2 and 3 are unpredictable.

When control is passed to the exit routine as a result of these two entry codes, execution is not in cross-memory mode. For online systems, execution is in the control region address space or, if a DL/I separate address space is used (LSO=S), execution is in the DL/I separate address space.

Restrictions

Keep the following restrictions in mind when using the segment edit/compression Facility:

- Because this routine becomes a part of the IMS control or batch region, any abnormal termination of this routine terminates the entire IMS region. Any user-written segment edit/compression exit routine should return to IMS with an abend code and a reason code instead of initiating a standard abend.
- The exit routine cannot use operating system macros such as LOAD, GETMAIN, SPIE, or STAE.
- All editing or compression of segments occurs as the segments are described in a physical database only. For specific restrictions, see *IMS Version 15.2 Database Administration*.
- The exit routine must not modify or alter the relative position of a key field in a DEDB segment. If the key field in a DEDB segment changes or moves during a compress or expand call, IMS issues abend 0799, subcode 1. For more information about this abend, see *IMS Version 15.2 Messages and Codes, Volume 3: IMS Abend Codes*.
- When you specify the maximum size of the data portion of the segment in the DBD, if you use the segment edit/compression exit routine with full-function variable-length segments, you might need to include extra bytes. These extra bytes are needed if your exit routine makes the segment larger than its maximum size. For example, if the maximum length of your data is 100 bytes and your exit routine might add 2 bytes to the segment, specify 102 bytes as the maximum size. Increasing the maximum size accounts for the size of the segment from the application program (100 bytes) and the 2 bytes added by the exit routine. This restriction does not apply to full function fixed-length segments or to

segments in DEDBs. Using the segment edit/compression exit routine for both types of segments might increase their data sizes to values that are larger than those specified in the DBD.

Communicating with IMS

All IMS control blocks provided to the segment edit/compression exit routine are for reference only; no data can be changed, including the segment at the source area address. The only modification allowed is the alteration of the segment during the move operation from the source to the destination address. DSECT addressability to the control blocks is provided by the IMS IDLI macro.

Contents of registers on entry

On entry to the exit routine, the registers contain the following:

Register	Contents	
0	Set to zero before call to exit routine. Can contain Abend code U2990 on return if the exit routine detected an error.	
1	Address of the Partition Specification Table (PST).	
2	Address of the first byte of the segment to be modified (source address).	
3	Address where the modified segment is returned (destination address). For DEDB segments, this area is 10 bytes larger than the maximum segment size. For full-function fixed-length segments, this area is 10 bytes larger than the maximum segment size, unless a larger size was specified in the DBD. For full-function variable-length segments, this area is the maximum segment size.	
4	Address of the physical segment description block (PSDB). From this block, the field description blocks (FDB) can be located. (Register 4 is always zero when a DEDB is accessed by the exit routine, because the PSDB does not exist for DEDBs.)	
5	Address of the segment edit/compression control section.	
6	Entry code (detailed in the following section):	
	0	Segment compression call
	4	Entire segment expansion call
	8	Partial segment expansion call (full-function databases only)
	12	Full-function database or DEDB area open call
	16	Full-function database or DEDB area close call
7	For DEDB only, the minimum length as coded in DBD (SDBLMIN). Register 7 is only valid for function code 0 (segment compression) and function code 4 (segment expansion).	
13	Address of save area. The exit routine must not change the first three words.	
14	Return address to IMS.	
15	Entry point of exit routine.	

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers.

Compressing and expanding segments

The following two entry codes are required for segment compression and expansion; they are used when you specify the DATA compression operand.

Code	Description
0	Segment compression call. The source address points to an uncompressed segment image as it appears in the application program input/output area.
4	Entire segment expansion call. The source address points to a compressed segment. Application program requests qualified on a data field require the use of entry code 4 for normal retrieval expansions.

To reduce the amount of processing overhead required with the movement of data, the following third entry is required when the KEY compression operand is used. The KEY operand is for use with full-function databases only. Key compression is not supported for DEDBs.

Code	Description
8	Partial segment expansion call with the KEY operand (full-function databases only). Expansion takes place from the start of the segment through the sequence field. This facility is required if you elect to use key compression, or if you compress any field that alters the starting position of the key field. All DL/I calls using sequence field qualification on key compressed segments require the use of this entry code.

Description of entry codes

The entry code that is passed to the exit routine in register 6 indicates the reason IMS called the exit routine. The five possible entry codes are described in the following sections.

Description of parameters

The length of the segment to be moved is provided in one of two places:

1. If the segment length specified in the DBD is a fixed length, the source length is in the DMBCPSGL field.
2. If the segment is defined as variable in length, the source length is provided as a binary value in the first two bytes at the source address.

In either case, the move operation provided by the edit/compression routine must result in a 2-byte length field, followed by the corresponding quantity of data in the segment work area.

To help you provide parameters to the edit/compression routine, the DBD control block has a table appended to it that is made up of assembly language control sections. One control section is developed for each segment type to be edited or compressed. Each control section has a CSECT name equal to that of the segment name.

These control sections are placed at the end of the DBD module. They contain information such as the segment edit/compression routine name, the name of the segment, and the total length of that control section. Each control section can be extended to contain any desired data or algorithm information. A sample segment control section is shown in the following table.

Table 34. Segment edit/compression control section (DMBCPAC)

Hex offset	Contents
+0	Segment name
+8	Routine name

Table 34. Segment edit/compression control section (DMBCPAC) (continued)

Hex offset	Contents				
+10	Entry point address		Flag byte	Sequence field length -1	Sequence field offset
+18	Segment length / maximum length	Total length of CSECT	Reserved for exit routine		
+20	Any user data (length varies)				

Information in the various fields shown in the previous code sample are as follows:

```
DMBCPAC    DSECT
DMBCPCNM   DS      CL8     Segment name
DMBCPCSG   DS      CL8     edit/compression routine name
DMBCPEP    DS      A       Entry point address
DMBCPFLG   DS      XL1     Flag byte
DMBCPKEY   EQU     X'02'   Segment has key compression
                           option
DMBCPNIT   EQU     X'01'   Initialization processing is
                           required
DMBCPVLR   EQU     X'04'   Segment is variable-length
DMBCPSEQ   EQU     X'08'   Segment has key sequence field
                           defined
DMBCPJJD   EQU     X'10'   Exit caller requests a return code
                           rather than hard abending.
DMBCPSQF   DS      XL1     Executable length of sequence
                           field, if defined
DMBCPSQL   DS      H       Sequence field offset
DMBCPSGL   DS      H       For fixed length segments -
                           segment length; for variable
                           length segments - maximum
                           length
DMBCPLNG   DS      H       Total length of CSECT; fixed
                           length plus length of
                           user-defined parameters (always
                           a multiple of 8)
DMBCPUSR   DS      0F      Any quantity of user-defined
                           data.
```

The first 28 bytes are constants defined by DBDGEN. When the new table is defined to include additional parameters, these fields must be duplicated. The only exception to this rule is that the CSECT length field must be updated to reflect the new length. After an assembly of the new table, bind is done to exchange the new table for the old one. User-added code should not contain address constants, because this CSECT is moved after it is loaded. Use an ENTRY statement to specify the name of the DBD when this operation takes place, as well as an ORDER statement to ensure that the original order of multiple CSECTs is maintained. For details about this, see the section on automatic CSECT replacement in the z/OS product library.

If your exit routine references IMS control blocks other than the one shown in Table 34 on page 121, you need to reassemble the routine using the current release of IMS.

Related reference

“Initialization of IMS callable services (DFSCSII0)” on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Description of sample segment compression/expansion modules

Use the sample segment compression/expansion modules to compress three, four or more repeated strings.

Subsections:

About this routine

Compression/expansion examples are provided as guidance to the IMS system user.

DFSCMPX0 and DFSKMPX0 can be used by either full-function databases or DEDBs. Both routines perform segment compression. The only differences are:

- DFSCMPX0 compresses three or more repeated strings. This exit routine employs z/OS services to accomplish segment compression and expansion. For more information on these services, see the z/OS library. (DFSCMPX0 is the recommended compression routine.)
- DFSKMPX0 compresses four or more repeated strings. This exit routine relies on programming logic to accomplish segment compression and expansion. (DFSKMPX0 is not recommended, but it will continue to be supplied and supported for compatibility reasons.)

When control is given to DFSCMPX0 or DFSKMPX0, the program checks the entry code passed in register 6. The entry code indicates whether the request is for compression of a segment or for the partial (full-function databases only) or entire expansion of a compressed segment. It then branches to an appropriate routine to perform the required task. On normal completion of the task, it returns control to the IMS Control Program with a return code of 0.

Specific rules and restrictions followed in compression and expansion of a segment are detailed in this topic. For sample code, see the IMS.SDFSSMPL library.

For the latest versions of DFSCMPX0 and DFSKMPX0, see the IMS.SDFSSMPL library; the member names are DFSCMPX0 and DFSKMPX0. Because DFSCMPX0 provides improved performance and possibly better compression, IBM does not recommend the use of DFSKMPX0.

The compression routine

Compression of a segment requires different data handling according to the data organization of the segment. The two data formats are fixed and variable-length.

You can specify the KEY (full-function databases only) or DATA operand for either of the two data formats. The following figure shows data before and after compression for both fixed- and variable-length segments.

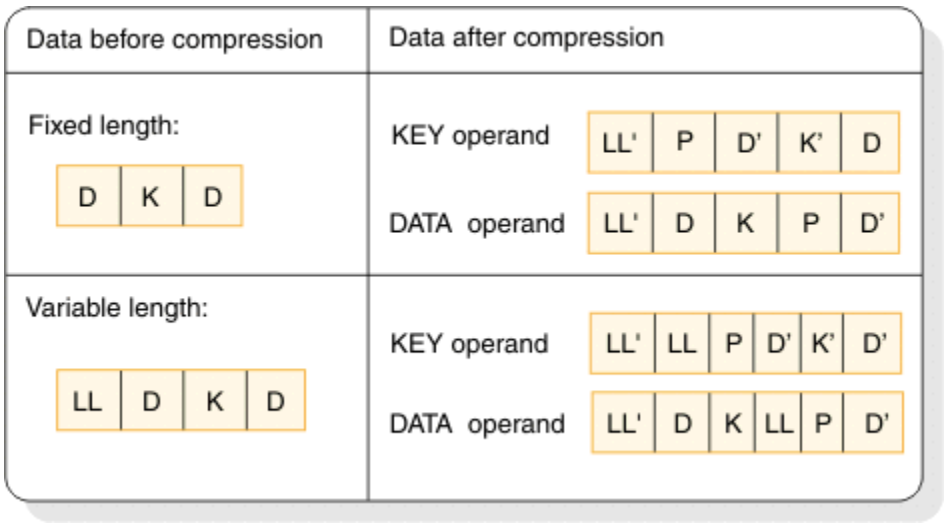


Figure 8. Data handling formats

D
data

K
pointer to the 1st CCB

LL'
new statement

LL
original segment length

D' and K'
compressed data and key

Compression of a segment results in one of the four formats listed in the preceding figure, depending on the original record format and the operand specified.

The initialization processing routine

When specified, IMS gives control to the segment edit/compression routine immediately after the databases are opened and immediately before the databases are closed.

When a command code is given to branch to the initialization processing routine or to the termination processing routine in the DFSKMPX0 program, the DFSKMPX0 program returns to the calling program. No processing of particular data is attempted at this stage.

Program messages and codes

When a Segment Edit/Compression exit routine detects a problem and initiates a standard abend, that abend can bring down the IMS. This severely impacts all other IMS applications running in an online IMS environment. The Segment Edit/Compression exit routines return to the caller with an abend code in register 0 and a reason code in register 15. Thus, abends in Segment Edit/Compression exit routines are converted to IMS abend U0840s so that only the dependent region that the abending application is running in is brought down.

The following table lists the abend codes.

Table 35. Program messages and codes - abend codes

User abend	Description
2989	A segment data organization is variable-length, but its length field is 2>N>32767 A fixed-length record, but the segment length in Compaction Control Table indicates: 0>N>32767
2990	A command code passed by the control program is out of a valid range: 0>N>16 1. REASON - D4D7E701: During a compression request, the input length of the variable length segment is less than 2 bytes. 2. REASON - D4D7E702: During an expansion request, the input length of the compressed segment is less than 2 bytes. 3. REASON - D4D7E703: During an expansion request, a non-zero return code was returned by the z/OS expansion service. (CSRCESRV). 4. REASON - D4D7E704: INIT was not specified in the COMPRTN parameter of the SEGM statement. 5. REASON - D4D7E705: Invalid function code. A command code passed by the control program is out of valid range. 6. REASON - D4D7E706: The key field length (sequence field) plus the offset of the key field within the segment is greater than the segment length indicated in the segment length field of a Compression Control Table. 7. REASON - D4D7E707: The length of a segment indicated in the segment length field of a Compression Control Table is negative.
2991	A command code is passed to compress after, or expand up to, a sequence field of a segment. No sequence field is defined in the segment.
2992	Any of the following conditions results in an abend with this code. Applicable to both fixed- and variable-length segments: • A D/K length is greater than an SCL length of a segment. Applicable only to a variable-length segment: • A D/K length is greater than a LL length. • A LL length is greater than an SGL length. • A LL length is less than 2. • An SGL length is less than 2. Applicable to a fixed segment: • An SGL length is a negative value. **D/K length =** A sum of length from the beginning of a segment to the end of a key field (SEQUENCE FIELD). **SGL length =** A length of a segment indicated in the segment length field of a Compression Control Table. **LL length =** A length of a variable-length record indicated in the first two bytes of a precompressed segment.

Program assumptions

All parameters and data passed by the IMS control program, such as the address of the input segment data, the output data area address, and the length of an input segment, are considered valid data.

The IMS control program passes an address of an input segment data area in register 2 and an address of an output data area in register 3.

The size of output data area is:

- A segment length plus two bytes for a fixed-length segment.
- The maximum segment length for a variable-length segment.
- No segment length greater than 32,767 bytes.

All segments processed by the compression routine are treated as variable-length by the IMS system control program, regardless of their precompression format.

Although no DFSKMPX0 sample exit routine is provided here, the exit routine is supported and supplied in the IMS.ADFSMPL library.

Hardware data compression support

You can compress or expand full-function and DEDB databases by using Hardware Data Compression support.

Hardware Data Compression (HDC) reduces DASD storage requirements for databases, reduces database I/O, and improves database performance.

With HDC support, you can generate exit routines to activate the hardware-assisted data compression available on processors. The processors use a compression technique that uses a fixed number of bits to replace a variable number of bytes.

If compression hardware is installed, the segment is compressed or expanded using the hardware instruction CMPSC. If compression hardware is not installed, the standard HDC exit routine calls the z/OS CSRCMPSC macro to compress or expand the segment by activating software simulation.

HDC compresses and expands segment data by calling a compression exit routine that has been specified on the SEGM statement during DBDGEN. This exit routine is created by binding a user-defined dictionary and an IMS-supplied base exit routine.

The space saved by compression depends on the user-defined dictionary, which performs the translation between compressed and uncompressed data. Different dictionaries are built for different sets of data. You receive the best results by creating a dictionary that compresses the most frequently occurring data in the largest databases.

If a fixed or variable-length segment requires compression and the data format is such that compression cannot take place, then the exit routine adds control information which indicates that the segment could not be compressed. This addition of the control information will lengthen the segment beyond the maximum length definition. To allow for this expansion and to allow IMS to validity check the compression results, you can add an arbitrary value of 10 bytes to the segment length.

If the segment length specified in the DBD is variable and the database is a DEDB, the length can exceed the maximum by up to 10 bytes but must not exceed 120 bytes less than the control interval (CI) size. If the segment length specified in the DBD is variable and the database is a HIDAM, HISAM, HDAM, or PHDAM the length cannot exceed the DBDGEN maximum.

Implementing HDC support

Using the Hardware Data Compression Dictionary (HDCD) utility (DFSZLDU0), you can implement hardware compression, build a hardware compression dictionary, and compare hardware compression statistics.

To implement hardware data compression with HISAM, HIDAM, PHIDAM, HDAM, PHDAM, and DEDB databases, follow these steps:

1. Create an HDC dictionary, using the Hardware Data Compression Dictionary utility (DFSZLDU0).
2. Bind the HDC dictionary to an IMS-supplied base exit routine, which produces a segment edit/compression routine. The base module is about 1 KB and is bound with 64-KB dictionaries. Therefore, the user exit routines require slightly more than 64 KB of memory.
3. In the DBDGEN SEGM statement COMPRTN parameter, specify the newly created segment edit or compression routine and the INIT keyword. The name of the routine must not be the same as the DBDNAME.
4. Unload the database using the old DBD.
5. Create the new DBD specifying the new exit routine.
6. Reload the database using the new DBD. (A new DBD requires that you run ACBGEN.)

Building the HDC dictionary

To build the HDC dictionary, use a sequential variable-length file as input to the HDCD utility. This must be a QSAM file of a variable record format and contain uncompressed segments, which are used to build the dictionary. You can create this QSAM file with a user-written unload program, or with the HD Reorganization Unload utility (DFSURGU0). Use your own data analysis to determine what uncompressed segments to use. Use the QSAM data set with the procedure.

Exception: If you use a QSAM file created by the DFSURGU0 utility, the dictionary build process includes (will not ignore) the header and trailer records created by the DFSURGU0 utility. Also, the dictionary build process includes (will not ignore) the prefix added to each data segment by the DFSURGU0 utility.

Other HDCD utility functions

In addition to creating the HDC dictionary, the HDCD utility provides:

- Compression statistics program, which is generated from the QSAM input file or from an alternate file. By using an alternate file, you can compare statistics and evaluate the dictionary's effectiveness.

  The compression statistics program:

  – Calculates the potential storage savings percentage as follows:

    SAVINGS=(100-((average compressed segment size/average precompressed segment size)*100)).

    If the potential storage savings do not meet the HDCDCTL default parameter's criteria, a dictionary object file is not built.

  – Prints the following statistics:

    - HDCDCTL parameters.
    - Number of segments read.
    - Smallest precompressed and compressed segment sizes.
    - Largest precompressed and compressed segment sizes.
    - Average precompressed and compressed segment sizes.
    - Potential storage savings percentage.

    The value shown for either the smallest or largest uncompressed segment could represent the length of the DFSURGU0 utility header or trailer segment.

  – Produces data integrity validation option.

  – Produces an object file for the specific HDC dictionary, provided that the following compression criteria are met:

- Precompressed data matches expanded data if the data integrity validation option is specified.
- Potential storage savings exceed the user-specified minimum percentage.

Related reference

“Sample JCL procedure” on page 128
To build the hardware compression dictionary, you must create a QSAM data set containing uncompressed database segments that can be used with JCL procedures.

Sample JCL procedure

To build the hardware compression dictionary, you must create a QSAM data set containing uncompressed database segments that can be used with JCL procedures.

Use the QSAM data set with the following JCL procedure.

```
//HDCDBLD  PROC
//         HDCDNAM=DFSZHDCD,    /*USER SUP. DICT NAME,8 CHARS*/
//         QSAMIN='USER.QSAM',    /* INPUT QSAM FILE NAME    */
//         QSAMIT='USER.QSAMALT', /* ALTERNATE QSAM FILE NAME*/
//         DICTLIB='HDC.DICTLIB', /* DICTIONARY LOAD LIBRARY */
//         DICTNAM='DFSZHXYZ',    /* USER DICT. MEMBER NAME  */
//         CMPXIT='USER.COMPLIB', /* COMPRESSION EXIT LIBRARY*/
//         CMPMBR='CMPXIT01',     /* USER EXIT MEMBER NAME   */
//         RGN=2048K,
//         SYS2=,
//         SOUT=*,
//         UNIT=SYSDA,
//         VOLSER=,


//         CYL=TRK,PRIM=5,SEC=2,BLKSZ=3120
//*************************************************************
//* CREATE STATISTICS AND HDC DICTIONARY OBJECT FILE.      *
//*************************************************************


//HDCDGEN  EXEC PGM=DFSZLDU0,REGION=&RGN,PARM=&DICTNAM
//STEPLIB  DD   DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
//HDCDIN   DD   DSN=&QSAMIN,DISP=SHR;
//HDCDIT   DD   DSN=&QSAMIT,DISP=SHR;
//HDCDOUT  DD   DSN=IMS.&HDCDNAM.HDCDOBJ,
//         DISP=(,CATLG,DELETE),
//         UNIT=&UNIT,
//         SPACE=(&CYL,(&PRIM,&SEC),RLSE),
//         DCB=(LRECL=80,BLKSIZE=&BLKSZ,RECFM=FB)
//HDCDCTL  DD  DUMMY    /* 'DUMMY' USES DEFAULT PARMS */
//*
//*************************************************************
//* CREATE LOAD MODULE FROM DICTIONARY OBJECT TEXT DECK.  *
//*************************************************************


//LINK1    EXEC PGM=IEWL,COND=(0,NE),


//    PARM='SIZE=(180K,20K),RENT,REFR,NCAL,LET,XREF,LIST'
//SYSLMOD  DD  DSN=&DICTLIB(&DICTNAM),DISP=SHR
//SYSUT1   DD  UNIT=&UNIT,DISP=(,DELETE),
//             SPACE=(CYL,(10,1),RLSE)
//SYSPRINT DD  SYSOUT=&SOUT
//SYSLIN   DD  DSN=IMS.&HDCDNAM.HDCDOBJ,DISP=(OLD,DELETE,KEEP)
//*
//*************************************************************
//* THE USER COMPRESSION EXIT ROUTINE IS BUILT BY LINKING *
//* MODULE DFSZLDX0 AND THE HDC DICTIONARY TOGETHER. THE  *
//* THE HDC DICTIONARY MUST BE THE FIRST CSECT WITHIN THE *
//* USER EXIT ROUTINE AND ALSO BE ON A PAGE BOUNDARY.     *
//*************************************************************
//LINK2    EXEC PGM=IEWL,


//         PARM='SIZE=(180K,20K),RENT,REFR,NCAL,LET,XREF,LIST'
//SYSLMOD  DD  DSN=&CMPXIT(&CMPMBR),DISP=SHR
<litdata>
//SYSUT1   DD  UNIT=&UNIT,DISP=(,DELETE),
```

```
//             SPACE=(CYL,(10,1),RLSE)
//SYSPRINT DD  SYSOUT=&SOUT
//SDFSRESL   DD  DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//DICTLIB  DD  DSN=&DICTLIB,DISP=SHR;
//*************************************************************
//*THE FOLLOWING CONTROL STATEMENTS MUST BE IN THE ORDER AS *
//*  ILLUSTRATED.                                           *
//*                                                         *
//*  DFSZHXYZ: THE HDC DICTIONARY NAME FOR THE SEGMENT.     *
//* (&DICTNAM) THIS HAS TO BE CHANGED TO A FIXED   NAME OF  *
//*            DFSZHDCD SO THAT THE COMPRESSION EXIT DRIVER *
//*            CAN BE LINKED TO IT.                         *
//*                                                         *
//*  DFSZLDX0: THE COMPRESSION EXIT DRIVER ROUTINE.         *
//*                                                         *
//*  &CMPMBR: USER SPECIFIED COMPRESSION/EXPANSION EXIT     *
//*           ROUTINE NAME THAT IS USED ON THE              *
//*           SEGM COMPRTN= (&CMPMBR,DATA) DBD STATEMENT.   *
//*************************************************************
//SYSLIN  DD *
  CHANGE  &DICTNAM(DFSZHDCD)     (&DICTNAM) DICTIONARY NAME
  INCLUDE DICTLIB(&DICTNAM)      DICTIONARY MUST BE 1ST CSECT
  INCLUDE SDFSRESL(DFSZLDX0)       STANDARD COMPRESSION EXIT
  PAGE    DFSZHDCD
  ENTRY DFSZLDX0
  NAME &CMPMBR(R)                (&CMPMBR) COMPRESSION EXIT
/*
// PEND
```

Subsection:

- "DD name descriptions" on page 129

DD name descriptions

HDCDIN DD
The input sequential variable length data set that contains the IMS database segment data that you extracted.

HDCDIT DD
The input sequential variable length data set or an alternate file that is used to calculate the compression statistics.

HDCDOUT DD
Output HDC dictionary object deck. The z/OS format dictionary is built and converted into a bind compatible object deck for subsequent use in the dictionary link edit step.

SYSPRINT DD
Compression analysis statistics.

HDCDCTL DD
A data set containing the following control statements. The value specified for a control statement must conform to the rules described for each control statement. Code the value after the keyword for the control statement. Use a blank or a comma to separate control statements.

RECS=
The number of input records to be processed. The default is ALL. Specify a number between zero and 2147483647. If any number outside this range is specified, the default ALL is used.

PERC=
The percentage of storage savings to be realized. The default is 5 percent. One or two digits are allowed.

INTEG=
By specifying Y or N, this keyword checks or does not check the data integrity of compressed segments. The default is N.

Tips for hardware data compression

Hardware data compression (HDC) can help you save I/O and storage.

To decide whether to use HDC, run the HDCD utility and analyze the output statistics to determine how much storage and I/O savings you can achieve.

You might want to limit the use of HDC to one time per database, since its implementation requires an unload and reload of the database.

Recommendation: Evaluate all the segments in a database before implementing compression. If you use compression for multiple segment types, implement compression for all of them at the same time.

Because uniquely tailored dictionaries yield the most compression, you should use the dictionaries for high-volume segments to maximize savings.

You can create more generally-tailored dictionaries for other reasons. If you know the type of data in most segments, you can create dictionaries by using a sampling of similar data from many of those segments. For example, you might want general dictionaries for upper-case text, mixed-case text, numeric, alphabetic, and general mixed data. You can use these dictionaries for multiple segment types, eliminating the need to produce unique dictionaries for each segment type.

Compression usually saves I/O for sequential processing and can also save I/O for random processing. Typically, savings for random processing is realized with large database records, especially if the record is spread over multiple blocks or CIs. Compression can reduce the number of blocks or CIs that must be read to access a segment. This is likely to apply to twin chains of multiple blocks or CIs, even after reorganizations.

Return codes from the HDCD utility

The HDCD utility ends and issues one of five return codes.

The following return codes can be issued from the HDCD utility:

Code	Description
0	Utility ended successfully and issued the accompanying DFS1170I message.
4	Utility ended successfully and issued the accompanying DFS1171W message, but it did not build a dictionary because the requested storage savings percentage was not met.
8	Utility ended successfully and issued the accompanying DFS1172E message, but it did not build a dictionary because data integrity checks were detected between a source QSAM input record and its equivalent re-expanded record.
12	Utility ended unsuccessfully and issued the accompanying DFS1173W message, because z/OS CSRCMPSC is not installed on the machine.
16	Utility ended unsuccessfully and issued the accompanying DFS1174E message, because a logic error occurred during invocation of the CSRCMPSC compression service macro.

Related Reading: For more information about these messages, refer to *IMS Version 15.2 Messages and Codes, Volume 4: IMS Component Codes*.

Sequential Buffering Initialization exit routine (DFSSBUX0)

This exit routine can dynamically control the use of Sequential Buffering (SB) for online and batch IMS subsystems, as well as DBCTL.

Subsections:

About this routine

By using one of the five sample SB routines that IMS provides or one that you write, you can:

- Disallow the use of SB.
- Specify that SB be conditionally activated by default whenever IMS detects a sequential I/O pattern in batch or BMP regions.
- Change the IMS default values for the number of buffer sets in each SB buffer pool.

The SB exit routine (DFSSBUX0) is called before each application program or utility. This enables the exit routine to dynamically change SB options and parameters and dynamically control how your system uses SB.

The following table shows the attributes of the Sequential Buffering Initialization exit routine.

Table 36. Sequential Buffering initialization exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	You must name this exit routine DFSSBUX0.
Binding	After you compile and test your module, bind it into IMS.SDFSRESL, SYS1.LINKLIB, or into any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control, SAS, and batch regions.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS callable services.

Loading the routine

IMS loads the routine at IMS initialization time.

Considering performance

DFSSBUX0 is called frequently during the scheduling of MPPs and PSBs of CICS in a DBCTL environment. If you modify an SB sample routine or write your own routine, code it to minimize overhead during the call to the routine for these programs.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
10	Address of partition specification table (PST).
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address of IMS.
15	Entry point of exit routine.

Description of parameters

DFSSBUX0 receives the address of a parameter area in Register 1. This parameter area is mapped by the DFSSBUXP macro and contains:

- The region type (batch, BMP, MPP, Fast Path, DBCTL) in the SBPRMREG field.
- The job, program, and PSB names. (Exceptions: IMS utilities executed without a PSB have a DBD name instead of a PSB name.)
- The message classes of the message region (when running in an MPP region).
- The IMS default values for SB options and parameters.

The following paragraphs describe how DFSSBUX0 can change the default values of SB options in the SB parameter area. Each change applies only to the current application program or utility being invoked. The DSECT of the parameter area is presented at the end of the discussion.

Disallowing the use of SB

The **SBPRMPDI** bit determines whether the use of SB is disallowed. The default value for this bit is *off*. DFSSBUX0 can set this bit on, however, to disallow the use of SB and cause IMS to ignore any PSBGEN or SB control card requests to the contrary. You can set this bit during peak periods of online use to save real storage space, especially if your system's real-storage is already constrained.

Conditionally activating SB by default

The **SBPRMPAD** bit determines whether IMS conditionally activates SB by default. The default value for this bit is off. DFSSBUX0 can set this bit on, however, so that IMS samples I/O reference pattern statistics of batch and BMP application programs. If IMS detects both a sequential I/O pattern and a reasonable activity rate, IMS activates SB. This occurs only if PSBGEN and SB control cards provide no specifications to override this process.

Exception: Since statistic sampling has an initialization overhead each time an application program is scheduled, IMS does not support conditionally activating SB by default for MPPs, Fast Path regions, or CICS applications.

You might want to use DFSSBUX0 to conditionally activate SB by default in the following situations:

- To activate SB for specific batch and BMP programs and for IMS utilities by setting the bit according to the program, job, or PSB name for a program
- To always set the bit to activate SB for all BMP and batch programs and for utilities for z/OS systems that are not storage-constrained
- To set the bit depending on the time of day (for example, during night batch processing when most sequential applications are running and a lot of storage is available for buffering purposes)

Changing the number of SB buffer sets

The **SBPRMPNR** full word field specifies a default value for the number of buffer sets (BUFSETS) in each SB buffer pool. The default value for this field is 4. However, DFSSBUX0 can set this field to a value ranging from 1 to 25, inclusive. If this value is greater than 1, SB can anticipate the future database calls of a BMP or batch program by concurrently reading the next set of blocks while IMS is processing current database calls.

Recommendation: If your databases are well organized, set a default BUFSETS value of 2 or 3 to save virtual storage space. If your databases are poorly organized, however, you can set a default BUFSETS value of 6 or greater to increase the chance that what your application program or utility is looking for is already in a buffer set.

DFSSBUX0 can also change the default BUFSETS value based on the time of day. For example, you might want DFSSBUX0 to choose a small value for BUFSETS during daytime main online processing time and a larger value during night batch processing time.

The following DSECT describes the format of the SB parameter area:

```
SBPRMP   DSECT
*
SBPRMP1  EQU   *      *****   READ-ONLY INFO FOR EXIT
SBPRMJOB DC    CL8' '         JOBNAME
```

```
SBPRMPGM DC    CL8' '         PGM NAME (BLANK FOR CICS)
SBPRMPSB DC    CL8' '         PSB NAME
SBPRMCLA DC    CL4' '         IMS MESSAGE CLASSES
SBPRMREG DC    X'00'          REGION-TYPE
SBPRMRE1 EQU   1              ...BATCH (EXCLUSIVE CICS)
SBPRMRE2 EQU   2              ...CICS
SBPRMRE3 EQU   3              ...BMP
SBPRMRE4 EQU   4              ...MPP
SBPRMRE5 EQU   5              ...IFP (FAST PATH)
         DC    XL3'00'         RESERVED
*
         DS    0F
SBPRMP2  EQU   *      *****    MODIFIABLE SB PARMS FOR EXIT
SBPRMPNR DC    F'0'            NBR OF BUFFER-SETS
SBPRMPFL DC    X'00'           FLAGS
SBPRMPDI EQU   X'80'           ...DISALLOW USAGE OF SB
SBPRMPAD EQU   X'40'           ...CONDITIONAL SB ACTIVATION BY DEFAULT
*
SBPRMPL  EQU   *-SBPRMP       LENGTH OF PARAMETER AREA
```

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers.

Related concepts

"Guidelines for writing IMS exit routines" on page 3
Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

OSAM sequential buffering (Database Administration)

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample SB initialization routines

Use the sample SB initialization routines in present form, modify, or use as guidelines for writing your own SB routine.

IMS supplies five SB sample routines. The first module disallows the use of SB; the next four cause IMS to conditionally activate SB by default.

SB sample routines	Description
DFSSBU1	The sample Sequential Buffering (SB) exit routine disallows the use of SB. For the latest version of the DFSSBU1 source code, see the IMS.SDFSSMPL library.
DFSSBU2	This sample exit routine causes IMS to activate Sequential Buffering (SB) by default when IMS detects a sequential I/O reference pattern and reasonable activity rate. This exit routine can be used for DataRefresher IMS utilities that can benefit from SB in both batch and BMP regions. For the latest version of the DFSSBU2 source code, see the IMS.SDFSSMPL library.
DFSSBU3	This sample exit routine causes IMS to activate Sequential Buffering (SB) by default when it detects a sequential I/O reference pattern and reasonable activity rate. In batch regions, this applies to all application programs and utilities; in BMP regions, this applies to DataRefresher, as well as those IMS utilities that can benefit from SB. For the latest version of the DFSSBU3 source code, see the IMS.SDFSSMPL library.

SB sample routines	Description
DFSSBU4	This sample exit routine causes IMS to activate Sequential Buffering (SB) by default when it detects a sequential I/O reference pattern and reasonable activity rate. This applies to all application programs and utilities in both batch and BMP regions. For the latest version of the DFSSBU4 source code, see the IMS.SDFSSMPL library.
DFSSBU9	This sample exit routine either disallows the use of sequential buffering (SB) or causes IMS to activate SB by default based on specific times of day. The routine is coded as follows: • The time between 1100 hours and 1400 hours is the peak period for processing online transactions. During this time frame, SB is disallowed. • During the time between 0900 hours and 1100 hours, and 1400 hours and 1700 hours, SB is neither disallowed nor activated by default for batch and BMP regions. • The rest of the time, SB is conditionally activated by default for batch and BMP regions. For the latest version of the DFSSBU9 source code, see the IMS.SDFSSMPL library.

Chapter 3. Transaction Manager exit routines

Transaction Manager exit routines provide support for message processing, including specialized routing and editing of messages. Additional routines perform terminal functions, provide security, and facilitate sign on and sign off support.

2972/2980 Input edit routine (DFS29800)

The 2972/2980 Input Edit Routine processes each entered message segment after that message segment has been translated by IMS.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

An input edit routine is required to perform terminal-related functions inherent in the design of the 2972/2980 General Banking Terminal system. IMS passes control to the 2972/2980 Input Edit Routine to process each entered message segment after that message segment has been translated by IMS.

The 2972/2980 Input edit routine must perform the following functions:

1. Determine the IMS destination (SMB or CNT) of messages entered from a 2980 teller or administrative station.
2. Determine end-of-message of multisegment messages (by setting DECCSWST bit 7 to indicate EOM).
3. Reposition the entered data at the beginning of the input buffer for IMS processing. The entered segment must be in standard IMS input message format after edit processing; a two-byte length field is followed by the text.

In addition to performing the preceding required functions, the 2972/2980 Input edit routine can add input terminal status information to the entered segment, such as the presence or absence of a passbook or auditor key on the input terminal. The 2972/2980 Input edit routine can initiate retransmission of the last successfully transmitted message to a 2980 logical terminal by a return code to the calling routine.

The following table shows the attributes of the 2972/2980 Input Edit exit routine.

Table 37. 2972/2980 input edit exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFS29800.
Including the routine	Because the Input Edit Routine will be called directly by the IMS 2972/2980 device dependent module (DFSDN110), you must bind the input edit routine with the IMS control region nucleus.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB in register 9 for the DFSCSII0 call. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS callable services.
Sample routine location	No sample is provided.

Communicating with IMS

Familiarity with IMS terminal handling procedures and control blocks is required for a user to write an Input edit routine to interface with IMS routines in the IMS control region. Examination of these control blocks might be required, but modification of IMS control blocks by a user-written routine seriously endangers the integrity of the entire system.

Contents of registers on entry

On entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Length of input buffer.
1	Address of the input area.
2	Length of input data. (The length of the area pointed to in register 1.)
7	Address of CTB.
9	Address of CLB.
11	Base of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of exit routine.

The format of the data contained in the buffer pointed to by register 1 at entry to the exit routine is as follows:

1. 9 blanks
2. Terminal address
3. Entered text

   If the entered text is from a 2980-4, the first byte of the entry is the teller identification.

Contents of registers on exit

On return to IMS, all registers must be restored except for registers 2, 10, and 15, which must contain the following:

Register	Contents	
2	Data length after edit (a zero length signifies a no-data segment).	
10	The inputting CNT address if a retransmission of the last successfully outputted message is required.	
15	One of the following return codes:	
	Return code	**Meaning**
	0	Process the entered segment.
	4	Re-send the last message to the CNT in register 10.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“Initialization of IMS callable services (DFSCSII0)” on page 16

Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

4701 Transaction Input Edit routine (DFS36010)

The 4701 Transaction Input Edit routine appends a blank and the eight-byte node name to a transaction input message. The routine also allows MPP to set up the appropriate change call for output.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

This exit is provided as a sample routine that appends a blank and the eight-byte node name to a transaction input message. If you have established a naming convention that relates node names to LTERM names, the node name can be used by the MPP to set up the appropriate change call for output.

The following table shows the attributes of the 4701 Transaction Input Edit routine.

Table 38. 4701 transaction input edit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFS36010.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSI00 call. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS callable services.
Sample routine location	IMS.ADFSSMPL (member name DFS36010).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of the input buffer
7	Address of CTB
9	Address of CLB
11	Address of SCD
13	Address of save area
15	Entry point of exit routine

Contents of registers on exit

On return to IMS, all registers must be restored except for register 15, which must contain the following return code:

Return code	Meaning
0	Normal processing

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

BSEX: Build Security Environment user exit (DFSBSEX0 and other BSEX exits)

The Build Security Environment user exit provides users with a mechanism to tell IMS whether or not to build the RACF® or equivalent security environment in an IMS dependent region for an application that has received its input message from neither OTMA nor an LU 6.2 device.

Use the Build Security Environment user exit to tell IMS whether to build the RACF® or equivalent security environment in an IMS dependent region for an application that has not received its input message from OTMA or from an LU 6.2 device.

You can also use this user exit to request that IMS bypass some part of the security processing in the dependent region when one of the following events occurs for a message that did not originate from an OTMA or LU6.2 device:

- CHNG call.
- AUTH call.
- Deferred conversational program switch on the local system where the inputting terminal is active. Security authorization for the deferred conversational program switch occurs only on the local system.

Subsections:

About this routine

The Build Security Environment user exit receives control before the first or next input message is given to an IMS application program and the input message is from neither OTMA nor an LU 6.2 device.

This routine executes in key 7, non-cross-memory mode under the dependent region TCB.

The following table shows the attributes of the Build Security Environment user exit.

Table 39. Build security environment user exit attributes

Attribute	Description
IMS environments	DB/DC, DCCTL. **Note:** Also supported in a DBCTL environment for non-message driven BMPs.

Table 39. Build security environment user exit attributes (continued)

Attribute	Description
Naming convention	You can name this exit routine DFSBSEX0 and link it into a library that is included in the STEPLIB concatenation. If DFSBSEX0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSBSEX0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDF*xxx* member of the IMS.PROCLIB data set. The routines are called in the order they are listed in the parameter.

Binding	You must write this user exit using reentrant coding techniques. You must link your user exit into the IMS.SDFSRESL library.
	
	If you use IMS callable services, you must link DFSCSI00 with your user exit. The following is an example of the bind JCL statements needed:
	
	`INCLUDE LOAD(DFSBSEX0)`
	`INCLUDE LOAD(DFSCSI00)`
	`ENTRY   DFSBSEX0`
	`NAME    DFSBSEX0(R)`

Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSBSEX0. If you use multiple exit routines, specify EXITDEF=(TYPE= BSEX,EXIT=(exit_names)) in the EXITDEF parameter of the USER_EXITS section of the DFSDF*xxx* member of the IMS.PROCLIB data set.
IMS callable services	To use IMS callable services with this user exit, examine the value of the SXPLATOK field in the "IMS standard user exit parameter list" on page 5: • If SXPLATOK is zero, you cannot use IMS callable services with this user exit. • If SXPLATOK is non-zero, the value is the callable services token for this user exit. You can use the 256-byte work area addressed by the SXPLAWRK field to call DFSCSIF0.
Sample routine location	No sample exit routine is provided.

Communicating with IMS

IMS uses the entry registers, the Standard User exit parameter list (SXPL), and the Build Security Environment user exit (BSEX) parameter list to communicate with this routine.

This routine uses register 15 to communicate with IMS.

Contents of registers on entry

The contents of the registers on entry are as follows:

Register	Contents
Register	Contents
1	Address of the IMS Standard User exit parameter list (SXPL).

Register	Contents
13	Address of a single standard z/OS save area.
14	Return address to IMS.
15	Address of BSEX.

All other registers are undefined.

Contents of registers on exit

The contents of the registers on exit are as follows:

Register	Contents
15	Return code indicating requested action: **Return Code (decimal)** **Meaning** **00** IMS is not to build the security environment during the scheduling phase of the transaction. The security environment can be built later if needed for processing a CHNG call, AUTH call, or a deferred conversational program switch. **04** IMS is to build the security environment during the scheduling phase of the transaction. If the security environment is needed later by a CHNG call, AUTH call, or a deferred conversational program switch, this same security environment is used. If the application program does not ever need the security environment, the build of the security environment is unnecessary. **08** Invoke the SAF interface (RACF, or equivalent product) on a CHNG call, an AUTH call, and a deferred conversational program switch, but bypass the dynamic creation of the security environment. If the transaction is running in the local system, and the user who entered the transaction is still signed on, the security environment created by SIGNON is used. Otherwise, the default security environment of the IMS control region or the IMS dependent region is used for the SAF call. Normally, the security environment of the dependent region is used. However, if the dependent region is running with LSO=Y or is a BMP with PARDLI=1 specified, then the security environment of the Control Region is used. **12** Bypass invoking the SAF interface on a CHNG call, an AUTH call, and a deferred conversational program switch. **16** Bypass invoking the SAF interface on a CHNG call, an AUTH call, and a deferred conversational program switch, and bypass the calls to the DFSCTRN0 and DFSCTSE0 user exits. **20** Invoke the SAF interface on a CHNG call, an AUTH call, and deferred conversational program switch, and bypass the calls to the DFSCTRN0 and DFSCTSE0 user exits.

Note:

1. For return codes 08, 12 and 16, IMS does not dynamically build the security environment during transaction scheduling, or later for a CHNG call, an AUTH call, or a deferred conversational program switch.

2. When return code 16 is used, the application gets a status code in the IOPCB of blanks. For the AUTH call, the status field in the I/O area has the value 24 (X'18'): transaction authorization not active.

All other registers are to be restored by this routine.

“IMS standard user exit parameter list” on page 5

This user exit uses the Version 6 standard exit parameter list. The address of the work area passed to this user exit in SXPLAWRK can be different each time that this user exit is called.

If your BSEX user exit can be called in an enhanced user exit environment, additional user exit routines might be called after your routine. When your user exit routine finds a transaction upon which to act, it can set SXPL_CALLNXTN in the byte that SXPLCNXT points to. This tells IMS to not call additional exit routines.

Build Security Environment user exit (BSEX) parameter list

The address of the BSEX parameter list (mapped by DFSBSEXP) on entry to this routine is contained in field SXPLFSPL of the IMS Standard User Exit parameter list. The following table describes the BSEX parameter list.

Table 40. BSEX parameter list (mapped by DFSBSEX0)

Offset	Field length	Description
X'00'	4 bytes	Transaction scheduling class.
X'04'	8 bytes	Transaction code of the input transaction.
X'0C'	8 bytes	PSB name.
X'14'	8 bytes	Program name.
X'1C'	8 bytes	User ID. Specifies one of the following: • Actual user ID of the user who entered the transaction. • LTERM name of the terminal from which the transaction was entered. • Blanks. This is the user ID for which the security environment will be built if requested by this exit routine.
X'24'	8 bytes	Group name.
X'2C'	32 bytes	Application parameter (APARM= on dependent region JCL).
X'4C'	64 bytes	First 64 bytes of the input message or zeros if the input transaction is conversational.
X'8C'	8 bytes	User ID of the dependent region address space.
X'94'	1 byte	Indicator for contents of user ID field: **U** User ID **L** LTERM **P** PSB name **O** Other name
X'95'	3 bytes	Reserved.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"RASE: Resource Access Security user exit (DFSRAS00 and other RASE exits)" on page 427
The Resource Access Security user exit (RASE) authorizes IMS resources such as transactions, PSBs, or output LTERM names. This user exit is called after the SAF interface is called.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Conversational Abnormal Termination exit routine (DFSCONE0)

The Conversational Abnormal Termination exit routine (DFSCONE0) provides an application program to clean up, if required, when a conversation is prematurely terminated.

This topic contains Product-sensitive Programming Interface information.

A conversational process terminates abnormally when:

- A conversation is ended by an /EXIT or /START command.
- A conversational application program terminates abnormally during a conversation.
- A conversational program fails to insert a message into a response PCB or into an alternate PCB that represents another conversational program.
- A non-correctable IMS conversational error occurs.

If used, the Conversational Abnormal Termination exit routine can be scheduled twice: once when an /EXIT or /START command is issued, and again either when an application program inserts a SPA, or when the conversational response is received from a remote system.

Subsections:

About this routine

You can provide an application program to clean up, if required, when a conversation is prematurely terminated. On entry, this program's I/O PCB contains the name of the terminal that had its conversation abended. An exit routine to schedule the application program is required. IMS provides a sample exit routine named DFSCONE0, or you can write your own. To use the IMS-provided routine, you must:

- Define a transaction code named DFSCONE.
- Write a nonconversational application program to be activated by DFSCONE.

When the sample exit routine (DFSCONE0) is finished, the IMS conversational processor determines whether the transaction DFSCONE has been defined. If DFSCONE is not defined, the conversation terminates and the SPA is discarded. If DFSCONE is defined, the conversational processor schedules the transaction DFSCONE with the SPA of the terminated conversation as a nonconversational single-segment message.

As an alternative, you can provide a more tailored exit routine. For example, you might want to interrogate the conversation control block (CCB) to determine the transaction that was in process when the conversation terminated, or you might want to inspect the SPA to find out what had occurred before the conversation terminated. No DL/I calls can be issued by your exit routine. A message processing program

should be scheduled to handle database inquiries and updates or extensive analysis of the conversation. The application program can send messages to the terminal associated with the terminated conversation.

To cause your application program to be scheduled, your exit routine must:

- Place the 8-byte name of the nonconversational transaction into the SPA (offset 6 bytes into the SPA).
- Set the desired length of the SPA.
- Insert information to be communicated to the scheduled program into the SPA.
- Set a return code of X'10' in register 15.

The transaction code inserted into the SPA must be for a valid, nonconversational transaction. Otherwise, no transaction will be scheduled, the SPA is discarded, and the response message (if available) is sent to the input terminal.

If you do not provide a DFSCONE0 exit routine, IMS processing is the same as if an exit routine existed and it returned a return code of 0. The default IMS action is as follows:

1. Terminate the conversation if it is still active.
2. Discard the SPA.
3. Discard the response message if available.

Attributes of the Routine

The following table shows the attributes for the Conversational Abnormal Termination exit routine.

Table 41. Conversational abnormal termination exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCONE0.

Binding	You must write this routine using reentrant coding techniques. You must link your routine into the IMS.SDFSRESL library.
	If you choose to use IMS callable services, you must link DFSCSI00 with your routine. The following is an example of the bind JCL statements needed:
	`INCLUDE LOAD(DFSCONE0)`
	`INCLUDE LOAD(DFSCSI00)`
	`ENTRY DFSCONE0`
	`NAME DFSCONE0(R)`

Including the routine	No special steps are required to include this routine. To use the sample user exit, you need to define the transaction DFSCONE.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function specific parameter list for the desired callable service. Use the ECB in Register 9 for IMS callable services.
Sample routine location	IMS.ADFSSMPL (member name DFSCONE0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

Register 0 contains a flag that identifies the reason why the conversation was terminated.

Byte	Contents
0	**Flags** **Meaning** **X'01'** Request for termination that is no longer active. **X'02'** The /EXIT or /START command was issued by a different terminal than the one in conversation; this causes the conversation to be terminated. If this flag is not on, the request for termination of the conversation is from the terminal in conversation. **X'04'** The input CNT could not be found. The master terminal of the current system is set as the input terminal. **X'08'** The transaction was discarded by the processing of the /EXIT command.
1	**Return Code** **Meaning** **X'01'** Conversation was terminated previously by an /EXIT, /START, or IMS cold start. The conversation transaction processed successfully, and IMS is sending (queuing) the response message to the input terminal.
2	Reserved

Byte	Contents
3	A flag byte that indicates the calling reason: **Flag** **Reason** **X'00'** Conversational application program abended. **X'04'** Reserved. **X'08'** /EXIT command for input or other (remote) terminal processed. **X'0C'** /START LINE or NODE command processed for terminal in conversation. The /START LINE command is valid only if no PTERMs are specified. **X'10'** SPA received for an inactive conversation. **X'14'** Inconsistent conversational definitions found in a multisystem conversation. Execute the /MSVERIFY command to show the inconsistencies. **X'18'** /EXIT command terminated the conversation and the latest SPA is not currently available. (It is queued for processing in this system, or it is in the MSC network.) The SPA passed to the exit routine is either the one from the previous step of the conversation, or a short SPA with just the header information. The exit routine is called with vector 10 when the current step in progress completes; at this time the latest (and last) SPA for the conversation is passed to the exit routine. This can not occur if an IMS restart results in the loss of the SPA in this or another IMS system. **X'1C'** The explanation for the /START LINE or NODE command is the same as for Vector 18. **X'20'** A conversational application program terminated without inserting to a response PCB or an alternate PCB that represents another conversational program. **X'28'** /EXIT command for input or other (remote) ISC terminal processed. **X'30'** The link receive entry point of the TM and MSC Message Routing and Control user exit routine (DFSMSCE0) canceled the input transaction.

The contents of the remaining registers are as follows:

Register	Contents
1	Address of the SPA.
2	Pointer to a parameter list that contains SPA processing options. See "SPA Options Parameter List" for a list of the parameters.
6	Address of the CCB for the terminal in conversation, if the conversation is still active. Zero if the conversation is already terminated.

Register	Contents
7	If zero, the conversation is already terminated. If positive, the register contains the address of the CTB for the terminal in conversation (if the conversation is active). If negative, the register contains the complemented address of the SPQB for the signed-off user, which can be the result of the exit being called because of an `/EXIT CONV USER` command.
09	Address of the ECB.
11	Address of the SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of DFSCONE0.

The following table shows the SPA options parameter list. This parameter list is mapped in the sample exit routine.

Table 42. SPA options parameter list

Field	Description
CONESPAH	Maximum SPA length
CONESPAL	Current SPA length
CONEFLG1	Flag 1. This flag can be set as follows: **CONE1TDO (X'80')** If this flag is set, register 1 points to a SPA buffer that contains the SPA at the maximum length. If this flag is not set, register 1 points to a SPA that is the length of the SPA for the current transaction. Truncated data option is set for the SPA parameter in the TRANSACT macro. **CONE1SQ (X'40')** shared queues are active.

Contents of registers on exit

On return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes:

Reason code	Meaning
X'00'	Exit has completed all cleanup required; no further action is necessary. IMS does the following: • Terminates the conversation (if still active). • Discards the SPA. • Discards the response message (if available).
X'04'	The conversation is ended. The name field is used as a transaction code for a new nonconversational transaction. The remaining data in the SPA is used as input data for a new transaction. IMS does the following: • Terminates the conversation (if still active). • Attempts to queue the SPA to the indicated transaction and schedule it. • Discards the response message (if available).

Reason code	Meaning
X'08'	Exit has completed all cleanup required. No further action is necessary. IMS does the following: • Terminates the conversation (if still active). • Discards the SPA. • Sends the response message to the input terminal (if available).
X'0C'	The conversation is ended. The name field is used as a transaction code for a new non-conversational transaction. The remaining data in the SPA is used as input for a new transaction. IMS does the following: • Terminates the conversation (if still active). • Attempts to queue the SPA to the indicated transaction and schedule it. • Sends the response message to the input terminal (if available).
X'10'	The conversation is ended. The name field is used as a transaction code for a new non-conversational transaction. The remaining data in the SPA is used as input data for a new transaction. IMS does the following: • Terminates the conversation (if still active). • Attempts to queue the SPA to the indicated transaction and schedule it. • Discards the response message (if available).

Notes for Contents of Registers on Exit:

1. If the SPA cannot be queued to the transaction because the transaction is not defined or defined incorrectly, the response message is still discarded.
2. On entry, if bit 7 in register 0, byte 1, is set on (R0='XX01XXXX'), the response message is available.
3. If the SPA cannot be queued to the transaction because the transaction is not defined or defined incorrectly, the response message is not discarded but is sent to the input terminal. On entry, if bit 7 in register 0, byte 1, is set on (R0='XX01XXXX'), the response message is available.

Related reference

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Destination Creation exit routine (DFSINSX0)

The Destination Creation exit routine creates an LTERM or a transaction when a destination for a message does not exist.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

IMS will call the Destination Creation exit routine to create an LTERM or a transaction when a destination for a message does not exist. DFSINSX0 tells IMS which type of destination to create: LTERMs, transactions for queuing, or transactions for scheduling. LTERM is the default destination.

The following table illustrates the types of destinations that are enabled under specific conditions that are specified for your environment in the IMS PROCLIB members:

Table 43. Environment specifications and destination types created by DFSINSX0

Environment specification:	Destination type:
ETO=Y	LTERM
SHAREDQ=*name*	Transaction for queuing
MODBLKS=DYN	Transaction for scheduling

Attributes of the routine

The following table shows the attributes of the Destination Creation exit routine.

Table 44. Destination Creation exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSINSX0.
Binding	This exit routine must be reentrant. The exit routine can be called in cross-memory mode.
Including the routine	If you want IMS to call the Destination Creation exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. If the exit routine is included, IMS automatically loads it.
IMS callable services	To use IMS callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found at offset 0 of the Destination Creation exit routine parameter list for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.ADFSSMPL (member name DFSINSX0).

Restrictions

The following restrictions apply to the use of the Destination Creation exit routine (DFSINSX0):

- DFSINSX0 is not called during XRF tracking on an XRF alternate system.
- When DFSINSX0 is used to create LTERMs, then DFSINSX0 and the Signon exit routine (DFSSGNX0) are corequisite. If you provide one exit routine to supply queue data for additional LTERMs, you must provide the other exit routine also. Both exit routines create the user control block structure and related LTERMs (including multiple LTERMs for a user): DFSINSX0 using an LTERM name and DFSSGNX0 using the user ID. These exit routines must contain the same logic so that the user structure is identical, regardless of which exit routine created it. However, DFSINSX0 cannot return the address

of a user descriptor. The address of a user descriptor can only be provided using the Signon exit routine (DFSSGNX0).

- When extended terminal option is inactive (ETO=N), you cannot write DFSINSX0 to create dynamic LTERMs. When ETO=N, you can write DFSINSX0 only to create dynamic transactions.
- When dynamic resource definition is disabled (MODBLKS=OLC) in the DFSCGxxx or the DFSDFxxx member of the IMS.PROCLIB members, you can write DFSINSX0 to create transactions that can only be used for queuing messages on the shared queues. You cannot write DFSINSX0 to create transactions that can be scheduled when dynamic resource definition is disabled.
- When shared queues are not active (the SHAREDQ= parameter is not specified on the IMS Procedure), you cannot use DFSINSX0 to supply destinations for queuing transactions.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
13	Save area address
14	Return address to IMS
15	Entry point address of exit routine

The following table shows the Destination Creation exit routine parameters. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

This parameter list is mapped by DSECT INSXMAIN, which can be found in the DFSINSXP macro.

Table 45. Destination creation exit parameter list

Offset	Length	Description
+0	4	ECB address.
+4	4	SCD address.
+8	4	User Table address.
+12	4	Address of a buffer for use by the exit routine to return user ID and queue data. The mapping of the buffer is DSECT USEQDATA in USEQDATA COPY. For additional details on the content and format, refer to the prolog in the sample routine (DFSINSX0 in IMS.SDFSSMPL). The value is zero for the following conditions: • An XRF alternate system. • The destination must be a transaction and there is an indicator at offset +20.

Table 45. Destination creation exit parameter list (continued)

Offset	Length	Description
+16	4	Address of a buffer for use by the exit routine to return Autologon Override parameters. The mapping of the buffer is DSECT ATLGPARM in DFSINSXP macro. For additional details on the content and format, refer to the prolog in the sample routine (DFSINSX0 in IMS.SDFSSMPL). The value is zero for the following conditions: • An XRF alternate system. • The destination must be a transaction and there is an indicator at offset +20.
+20	4	Address of a buffer containing destination name, and other environment flags, including indicators for the following: • Dynamic resource definition, ETO, or shared queues is enabled. • LTERMs or transaction control blocks can be created. • The exit routine output is an LTERM or a transaction control block. The mapping of the buffer is DSECT INSXDATA in DFSINSXP macro.
+24	4	Address of a buffer for use by the exit routine. The mapping of the buffer is DSECT INSXTRNQ in DFSINSXP macro. The buffer returns information that is used to create a transaction control block if the destination is a transaction, including transaction attributes. The value is zero if the destination is an LTERM.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains one of the following return codes. If an application INSERT call forced the LTERM creation, IMS ignores the return code.

Return code	Meaning
0	IMS creates the destination.
nonzero	IMS rejects the destination creation attempt for alternate PCBs. If an application INSERT call caused IMS to attempt the destination creation, the nonzero return code is returned to the application as an 'A1' status code. I/O PCBs force LTERM creation and ignore the return code.

In addition to the return codes, the exit routine can indicate whether to create an LTERM (set INSXTYPE equal to INSXCNT in the INSXDATA DSECT) or a transaction (set INSXTYPE equal to INSXSMB in the INSXDATA DSECT).

Related concepts

Remote LTERMs (Communications and Connections)

MSC descriptors (System Definition)

Related reference

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

"Signon exit routine (DFSSGNX0)" on page 281
IMS calls the Signon exit routine for signon processing if ETO=Y is specified as an execution parameter.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

DFSINSX0 when extended terminal option is active

When ETO=Y, you can write DFSINSX0 to supply queue data that will create local or remote LTERMS, when the destination does not exist.

You can specify that the extended terminal option is active by stating that ETO=Y in the IMS or DCC startup procedure.

Based on the selected user descriptor when ETO=Y, DFSINSX0 can perform the following tasks:

- If the selected user descriptor is the DFSUSER descriptor,
  - Add additional LTERMs to the structure and supply queue data for those additional LTERMs, based on supplied autologon parameters such as LU name, user ID, logon descriptor name, and mode table name.
  - Override queue data and autologon parameters.
  - Override the user ID derived from the user structure.
  - Provide the correct user ID for the user receiving messages.
  - Use the correct user ID to create the name of the user control block structure, including LTERM control blocks.
- If the selected user descriptor is a non-DFSUSER descriptor,
  - Override queue data and autologon parameters for only one LTERM that is derived from the non-DFSUSER descriptor.
  - Cannot override the user ID.

If no user ID is supplied and extended terminal option is active, the name of the user structure is the name of the target LTERM. If no user control block structure exists, IMS uses the same name for both the target LTERM and the selected user descriptor.

IMS creates LTERMs from information in the selected user descriptor, from information that the Destination Creation exit routine supplies, or, in the case of remote LTERMS, IMS will use Multiple Systems Coupling (MSC) descriptors. If an LTERM is not available (that is, it is already assigned to another user), the user control block is created without the LTERM. LTERMs can be added later using the `/ASSIGN` command.

Related reading:

- See *IMS Version 15.2 Communications and Connections* for more information regarding ETO.
- The Destination Creation exit routine creates destinations based on environmental specifications. For more information on these specifications, see the prolog of sample DFSINSX0.

Providing queue data and autologon parameters

Depending on the user descriptor selected, the Destination Creation exit routine can provide queue data (LTERM data) and autologon parameters. If the exit routine returns data that it is not allowed to return (as discussed in the following cases), IMS rejects the LTERM creation attempt.

There are two cases which describe what data the Destination Creation exit routine can supply. The two cases are based on whether a DFSUSER (Case 1) or a non-DFSUSER (Case 2) descriptor is selected. (For this exit routine, non-DFSUSER descriptors are descriptors based on the target LTERM name.) Each case is discussed in the sections that follow.

If the Destination Creation exit routine does not provide data to override the existing queue data, IMS proceeds as if you did not include the Destination Creation exit routine; IMS uses the information in the selected user descriptor to create the LTERMs.

Case 1

If the DFSUSER descriptor is selected, the Destination Creation exit routine:

- Can supply any of the fields defined in the interface (including LTERM names). The exit routine can change LTERM data, but not the actual name of the first LTERM provided.
- Can provide data for additional LTERMs.
- Can provide the correct user ID to override the user ID derived from the target LTERM.
- Can override autologon parameters. If the user structure already exists, the user's existing autologon parameters are not changed.

Case 2

If a non-DFSUSER descriptor is selected, the Destination Creation exit routine can only specify queue data to override data derived from the user descriptor. The exit routine:

- Can supply queue data (except LTERM names) to override data that the descriptor provides,
- Can override autologon parameters. If the user structure already exists, the user's existing autologon parameters are not changed.
- Cannot provide data for additional LTERMs or override the user ID.

In both cases, IMS verifies the additional LTERMs that are specified against the LTERMs that already exist in the system. IMS automatically allocates the user to the indicated node and attempts to establish a session with that node. If an LTERM that is specified as an additional LTERM already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user for which messages are queued.

Identifying which user descriptor IMS selected

If the user control block structure already exists for the user for whom messages need queuing but for which the target LTERM is missing, IMS selects the user descriptor that was used to build the user structure and calls the exit routine. If IMS locates the target LTERM name, it selects that user descriptor and calls DFSINSX0.

If IMS does not find a descriptor that contains the target LTERM name, it selects DFSUSER to create the user structure. IMS renames the descriptor, giving it the name of the target LTERM, and equates the user ID to this name. IMS then calls DFSINSX0, which can supply the correct user ID, overriding the one derived from the target LTERM.

If no user descriptor can be found, including DFSUSER, IMS rejects the LTERM creation request.

Remote LTERM creation for Multiple Systems Coupling

If Multiple Systems Coupling (MSC) is being used, the exit routine can request that a remote LTERM (RCNT) be built instead of a local ETO LTERM (CNT) if the destination of the message is an LTERM in a remote system. The exit routine supplies the associated MSC MSNAME and the remote LTERM name in

field INSXMSN in the INSXDATA input parameter list. This name is a link name (MSNAME) rather than a descriptor name.

The MSNAME and remote LTERM input creates the RCNT, similar to if an MSC descriptor had been used. Do not change any other parameter values in the INSXDATA input parameter list. The RCNT is assigned to the link name (LNB) representing the MSNAME.

Related Reading: For more information on MSC descriptors, see *IMS Version 15.2 System Definition*.

DFSINSX0 when shared queues are active

You can use the Destination Creation exit routine (DFSINSX0), formerly called the Output Creation exit routine, to create a transaction that queues messages in the shared message queues.

Before enabling dynamic resource definition or shared queues, evaluate any existing DFSINSX0 exit routines. The DFSINSX0 exit might need to be changed so that it checks whether LTERM creation is allowed before it accesses the USEQDATA parameter list that is related to LTERM processing. If LTERM creation is not allowed, the USEQDATA buffer address (INSXAUSQ) is zero.

If you specify that shared queues are active (SHAREDQ=*name*) in the IMS PROCLIB members, you can create a transaction that queues messages in the shared message queues and can be processed by another IMS in the IMSplex. The transaction cannot be scheduled on the local IMS system unless DRD is also enabled.

When the exit routine indicates that the destination is a transaction, IMS creates a transaction control block. DFSINSX0 returns information to IMS about the transaction, including whether the transaction is in conversational or response mode, and the SPA size if applicable. The transaction control block is not deleted until IMS is restarted. IMS can use the same transaction control block if it encounters additional instances of the undefined transaction input message.

The Destination Creation exit routine creates destinations based on environmental specifications. For more information about these specifications, see the prolog of sample DFSINSX0.

Related information

DFS3824 (Messages and Codes)

DFSINSX0 when dynamic resource definition is enabled

If dynamic resource definition is enabled, DFSINSX0 can create transactions that can be used for queuing messages and it can create transactions that can be scheduled. When DFSINSX0 creates transactions that can be scheduled, DFSINSX0 also has the ability to create programs for those transactions.

Before enabling dynamic resource definition or shared queues, evaluate any existing DFSINSX0 exit routines. The DFSINSX0 exit might need to be changed so that it checks whether LTERM creation is allowed before it accesses the USEQDATA parameter list that is related to LTERM processing. If LTERM creation is not allowed, the USEQDATA buffer address (INSXAUSQ) is zero.

If you specify that dynamic resource definition (DRD) is enabled (MODBLKS=DYN) in the IMS PROCLIB members, DFSINSX0 can create a transaction and an application program for scheduling on the local IMS system.

If inconsistent or invalid transaction attributes are returned, the new transaction is not created, and the message for that transaction code is rejected as an invalid destination. Any transactions or application programs created by DFSINSX0 inherit the global TRANSTAT parameter, as specified in the DFSDFxxx IMS.PROCLIB member.

Transactions created for scheduling

Transactions that are created for scheduling can be enqueued, scheduled, and executed. The DFSINSX0 exit routine can set attributes for the transaction and application program in the appropriate fields in the INSXTRNQ parameter list DSECT.

Transactions created only for queuing

Transactions that are created only for queuing by the DFSINSX0 exit routine have a status of DYN. The 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 a checkpoint has not yet occurred since creation of the transaction.

Before DFSINSX0 is called, you do not have to define the application program that is scheduled to process the transaction. If the application program is not already defined, DFSINSX0 can create the program with specific attributes. The DFSINSX0 exit routine can set the same attributes as those that are set by the **CREATE TRAN** command.

DFSINSX0 does not require Resource Manager (RM) to dynamically create a transaction. However, if RM is using a resource structure, and the transaction is created for queuing or scheduling in any IMS system, the new transaction name is registered with RM. This prevents another IMS system from creating an LTERM with the same name.

The Destination Creation exit routine (DFSINSX0) exit might fail with a completion code of 1D7 and the DFS3824 message if the default descriptor is being imported from the IMS change list in the IMSRSC repository or was not successfully imported from the change list. This error can occur if the default descriptor is not the IMS system-defined default descriptor.

Subsections:

- "Creating transactions across an IMSplex" on page 154
- "Creating default or duplicate transactions" on page 155
- "Exporting resource definitions to the IMSRSC repository" on page 155

Creating transactions across an IMSplex

DFSINSX0 exit routine can create transactions on other IMS systems in an IMSplex in specific environments. The following table lists these environments, and the options available to DFSINSX0 in these environments.

Table 46. Environments in which the DFSINSX0 exit routine can create transactions across an IMSplex

Environment	Options that the DFSINSX0 exit routine can use to create transactions
Non-shared queues	Dynamic transactions that the DFSINSX0 exit routine creates are always for scheduling. Bit TRNQ_FC_SCHD is ignored; however, if you set this bit, your exit does not need to be recoded if you move to a shared queues environment.
Shared queues, without the Structured Call Interface (SCI)	Dynamic transactions that the DFSINSX0 exit routine creates can be either for queuing (TRNQ_FC_SCHD = 0) or for scheduling (TRNQ_FC_SCHD = 1). The transaction is created on the local IMS system only (the system in which the DFSINSX0 exit routine is called). The dynamic transaction definition is not propagated to other IMS systems in the IMSplex.

Table 46. Environments in which the DFSINSX0 exit routine can create transactions across an IMSplex (continued)

Environment	Options that the DFSINSX0 exit routine can use to create transactions
Shared queues with SCI	Dynamic transactions that the DFSINSX0 exit routine creates can be either for queuing (TRNQ_FC_SCHD = 0) or for scheduling (TRNQ_FC_SCHD = 1). The transaction can be created for the following: **Queuing on the local IMS only** If TRNQ_FC_SCHD is set to 0, the transaction is created for queuing on the local IMS system only. Field TRNQ_IMS is ignored. This is the default if your exit does not modify bit TRNQ_FC_SCHD. **Scheduling on the local IMS only** If TRNQ_FC_SCHD is set to 1 and no name is set in field TRNQ_IMS, the transaction is created for scheduling on the local IMS. It is not created on any other IMS in the IMSplex. **Scheduling on one local IMS and one additional IMS, while queuing on all other IMS systems** If TRNQ_FC_SCHD is set to 1 and the name (IMSID) of an IMS is specified in the TRNQ_IMS field, a transaction is created for scheduling on both the local IMS and on the IMS whose IMSID is specified in the TRNQ_IMS field. If the IMSID specified in the TRNQ_IMS field refers to the local IMS, the transaction is created for scheduling on the local IMS only. In both cases, the transaction is created for queuing on the other active IMS systems in the IMSplex. If the transaction is already created for scheduling on one or more of the other IMS systems in the IMSplex, it will not be changed to a queuing-only transaction. The transaction will still be able to be scheduled on those IMS systems. **Scheduling on all IMS systems in the IMSplex** If TRNQ_FC_SCHD is set to 1 with an asterisk (*) in field TRNQ_IMS, the transaction is created for scheduling on all IMS systems that are currently active in the IMSplex.

Creating default or duplicate transactions

If you want the DFSINSX0 exit routine to create a transaction using the current set of system defaults (that is, as specified by the current transaction default descriptor), do not set any of the definition bits in the INSXTRNQ DSECT. If you want the DFSINSX0 exit routine to create a transaction that matches an existing transaction or descriptor, specify the name of the transaction or descriptor in the TRNQ_TRAND field of the INSXTRNQ DSECT. You may need to specify the program name if the descriptor does not have a program name defined.

If you create a transaction or program but specify an invalid attribute combination in the INSXTRNQ parameter list, you will receive message DFS3424I to help diagnose the problem. Message DFS3424I contains the resource name, return code, reason code, and completion code, if applicable.

Exporting resource definitions to the IMSRSC repository

The transaction and program resources that are created by DFSINSX0 can be defined to be exported by setting TRNQ_FC_EXPORT=1 on the exit parameter list. If IMS is defined to use the repository, the resources created by DFSINSX0 are exported to the repository when one of the following conditions is satisfied:

- The names of the resources are specified with the NAME keyword on the **EXPORT TARGET(REPO)** command
- An **EXPORT DEFN TARGET(REPO) OPTION(CHANGESONLY)** command is issued after DFSINSX0 creates the resources
- The resources are created in-between the range specified by the STARTTIME and ENDTIME keywords on the **EXPORT DEFN TARGET(REPO)** command

Related reading:

- The Destination Creation exit routine creates destinations based on environmental specifications. For more information about these specifications, see the prolog of the sample DFSINSX0 module in IMS.ADFSSMPL.

Related concepts

Monitoring transaction-level statistics (System Administration)

Dynamic resource definition (System Definition)

Related reference

EXPORT command (Commands)

CREATE TRAN command (Commands)

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

Fast Path Input Edit/Routing exit routine (DBFHAGU0)

The Fast Path Input Edit/Routing exit routine (DBFHAGU0) provides the minimum level of support required for IMS to use Fast Path's Expedited Message Handler (EMH).

IMS systems with a very high transaction rate use EMH. EMH is a performance option that speeds up message processing by imposing restrictions on message lengths and segmentation. To use EMH, an edit/routing routine must receive control from the Input exit routine and determine the eligibility of an incoming message for Fast Path processing. The sample exit provides the minimum level of support required to use IMS Fast Path.

Subsections:

About this routine

The Fast Path EMH buffer is dynamically allocated and might not be present at entry. Therefore, DBFHAGU0 can receive the message in an EMH buffer or queue buffer, depending on the terminal type. The exit routine is not permitted to move the data out of the input location. If the message is in a queue buffer at entry, the Fast Path system moves it to an EMH buffer. In editing the input message, the application should not increase the length beyond a length that fits in any message buffer.

If an EMH buffer cannot be obtained, the following message is sent to the input terminal:

```
DFS3971 Unable to process Fast Path due to EMH buffer shortage
```

The following table shows the attributes for the Fast Path Input Edit/Routing exit routine.

Table 47. Fast Path input edit/routing exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL
Naming convention	You must name this exit routine DBFHAGU0.

Table 47. Fast Path input edit/routing exit routine attributes (continued)

Attribute	Description
Binding	This exit routine must be reentrant if APPC/IMS support is active.
Including the routine	DBFHAGU0 is a separately linked module in the IMS.SDFSRESL. IMS automatically loads it during Fast Path initialization. If IMS cannot find DBFHAGU0, IMS terminates abnormally with ABENDU1011 and displays the following message: `DFS2730A UNABLE TO LOAD FP INPUT ROUTING EXIT: DBFHAGU0`
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found at offset X'0' of the Fast Path Input Edit/Routing Exit parameter list for the DFSCSII0 call. This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.SDFSSMPL (member name DBFHAGU0).

Expanding the routine

A transaction that is not Fast Path-exclusive can be directed to EMH processing by an expanded edit/routing routine, based on some condition or conditions beyond transaction code. For example, certain transactions can be routed to EMH if they originate at specified physical or logical terminals or if they reference the content of some portion of the message (for example, account number). The user-supplied DBFHAGU0 would have to develop appropriate routing codes based on such conditions.

Using the routine with shared EMH queues

If your installation uses shared EMH queues, DBFHAGU0 can place messages on the shared-queue structure for processing by any sharing IMS subsystem in the sysplex.

You can modify the exit routine to specify an application name for the application program used to process Fast Path input messages. If you do not specify an application name, Fast Path locates the transaction or routing code in the local IMS subsystem. Fast Path rejects the input message if it cannot locate the transaction or routing code.

You can also specify a sysplex processing code that determines how a message transaction or routing code is processed. The following sysplex routing options are available:

Local First
Specifies that the message is processed on the local subsystem if an IFP region is available. If no IFP region is available, the message is passed to the EMH queue structure. A program name specified in the exit routine for message processing overrides the transaction or routing code. Local First is the default.

Local Only
Specifies that Fast Path does not place the message on the EMH queue structure. Fast Path input messages are processed on the local IMS subsystem.

Global Only
Specifies that Fast Path places the input message on the EMH queue structure. The application program that processes the input message must be active on all sharing IMS subsystems. If the application is not active, Fast Path discards the input message and issues an error message. A program name specified in the exit routine for message processing overrides the transaction or routing code.

Recommendation: To avoid implicit priority for Local Only messages over Local First messages, process Local First and Local Only messages under separate program names. IMS places Local Only messages on the balancing group (BALG) queue and Local First messages on the shared EMH queue. When an IFP region becomes available, it checks the BALG queue for messages to process before it checks the shared EMH queue. This sequence gives implicit priority to Local Only messages that are processed in the same program.

Restrictions

You must rewrite your Fast Path Input Edit/Routing exit routine for this release of IMS, based on the DBFHAGU0 sample (located in the IMS.SDFSSMPL library) and the guidelines in this .

The exit routine cannot move the data out of the input location.

The exit routine must not increase the length of the message beyond a length that fits in any message buffer.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of Standard Exit Parameter List.
13	Save area address.
14	Return address to IMS.
15	Entry point address of exit routine.

This exit routine uses the Version 1 standard exit parameter list.

The following table lists the Fast Path exit parameters. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 48. Fast Path input edit/routing exit parameter list

Offset (decimal)	Length (decimal)	Description
+0	4	ECB address.
+4	4	SCD address.
+8	4	Input message.
+12	4	Address of routing code table entry if this is a Fast Path exclusive transaction, or zero.
+16	4	Eight-character work area to supply a routing code name.
+20	4	Address of ESCD.
+24	4	The length of the EMH Buffer for this application.
+28	4	Address of the DBFHAGU0 extended parameter list. This parameter list exists if shared EMH queues are used. Otherwise, the extended parameter list is 0.

The following table lists the Extended Parameter list parameters.

Table 49. DBFHAGU0 extended parameter list

Offset (decimal)	Length (decimal)	Description
+0	4	Address of the 8-byte PSB name
+4	4	Sysplex processing code (decimal) **0** Local First (Default) **4** Local Only **8** Global Only
+8	4	Address of the Local PSB name table
+12	4	Address of the Global PSB name table
+16	4	System definition code (decimal) **0** Transaction Defined in local system **4** Transaction not defined in local system
+20	4	Input message code (decimal) **0** Fast Path exclusive transaction **4** Fast Path potential transaction

Note:

1. The sample DSECT for the local program name table and the global program name table can be found in the DBFPGNT macro.

Contents of registers on exit

On return, all registers must be restored except for register 1 and 15, which must contain the following:

Register	Contents
1	Message number to send to inputting terminal.
15	One of the following return codes:

Register	Contents	
	Return code (decimal)	**Meaning**
	00	Schedule with Fast Path. Register 3 points to the RCTE to be used.
	04	Schedule with Fast Path using transaction code as the routing code.
	08	Schedule with Fast Path using the routing code you provide.
	12	Return to IMS for processing.
	16	Schedule with Fast Path using transaction code if the routing code equal to transaction code is active; otherwise, let IMS process it.
	20	Schedule with Fast Path using routing code provided the routing code is active; otherwise, let IMS process it. This is the same action as user exit return code 08.
	24	Discard input, send message from user table back to inputting terminal.
	28	Discard input, send message from system message table.

Related reference

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Front-End Switch exit routine (DFSFEBJ0)

The Front-End Switch (FES) exit routine allows you to keep the input terminal in response mode while it is waiting for the reply from the processing system for messages entered in an IMS system by a front-end switchable VTAM node and processed in another system (such as IMS or CICS).

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

You specify the FES exit routine in the IMS startup parameter with the FESEXIT parameter. You specify which static VTAM nodes can do front-end switching during system definition. You specify which dynamic VTAM nodes can do front-end switching in the ETO logon descriptors.

The connection between intermediate IMS systems must be through Intersystem Communication (ISC), although connections with non-IMS back-end systems can be any VTAM protocol that IMS supports, such as SLUTYPEP or SLUTYPE2. IMS-to-IMS and IMS-to-non-IMS interconnections are referred to as intermediate/back-end or IBE links, and front-end systems are referred to as FE systems.

Front-End Switch is not related to Multiple Systems Coupling (MSC), and cannot be used with MSC for the processing of the same transaction. Front-End Switch is designed to connect an IMS network to non-IMS systems, and MSC is used for homogeneous IMS networks.

Attributes of the routine

The following table shows the attributes of the Front-End Switch exit routine.

Table 50. Front-end switch exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSFEBJ0.
Binding	Prior to IMS 15, DFSFEBJ0 was linked into the IMS nucleus, which automatically made it non-reentrant (because the nucleus is non-reentrant). In IMS 15, DFSFEBJ0 was removed from the IMS nucleus and is now loaded as a stand-alone module during IMS initialization. Thus, if you use IMS 15 and you link this user exit as reentrant, you must ensure that it is not dependent on any information from a previous iteration and that it does not store into itself. To learn more, see Migration considerations for removing user exit routine specification from system definition (Release Planning) and Routine binding restrictions (Exit Routines).
Including the routine	If you want IMS to call the exit, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of IMS.SDFSRESL. If the exit routine is included, IMS automatically loads it each time IMS is initialized.
IMS callable services	To use IMS callable services with this routine, you need to issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in Register 9 for IMS callable services. Link DFSCSI00 with this exit manually to use callable services.
Sample routine location	IMS.ADFSSMPL.

You must code the exit routine for AMODE=31. You can define the RMODE as ANY.

Restrictions

The following restrictions apply to the Front-End Switch exit routine:

- The FES function can be used with the COMM macro statement specifying OPTIONS=BLKREQD or NOBLANK. However, you must specify a blank following the transaction code regardless of the option specified.
- If the back-end or intermediate system detects an error for an input transaction, the error message can not be sent back to the input terminal. It is sent to the MTO of the system detecting the error. It also can be sent back over the IBE session that sent the original input, or the input message can be sent to an ERP, if one is specified.

  If an error is sent over the ISC session, IMS will CLSDST the session thus making the error more visible and keep future ones from occurring. This can be valuable during a debugging period of a new FES exit or application; however, it can prove bothersome during production time. To avoid this, specify a FEIBERPN when processing input from an ISC session and develop an application to log or process these errors should they occur.
- Conversational transactions are not supported.

- If the front-end system is part of an XRF complex, the terminal operator might not get the reply to a switched message in case of a takeover even if the reply comes in time. The terminal receives an IMS message instead.
- For a local transaction defined as full-function, nonresponse mode, the exit routine switches a transaction (TXNA) to a local transaction (TXNB) and turns on the timer facility. TXNB executes locally and replies to the originating terminal. However, the terminal is left in response mode. When the timeout transaction processes, a response is sent to the terminal, which resets the response mode.
- If the back-end system is non-IMS, the reply message that the back-end system sends to IMS must be asynchronous (nonresponse) and expect no counter-response from IMS. You can do this in one of two ways:
  - End the response with an end bracket (EB).
  - Append the FMH6 SCHEDULER header to the FMH5 header at attach time, and use a change direction (CD) indicator.

Related Reading: For more information, see *IMS Version 15.2 Communications and Connections*.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the FEIB. The FEIB contains all the information necessary for the exit to function. The exit routine must store additional information in the FEIB which is required for successful processing.
9	Address of ECB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return code	Meaning
0	No message switching
2	New destination from FE
6	New destination from IBE
8	Reply message
12	User table error

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“Initialization of IMS callable services (DFSCSII0)” on page 16

Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Terminal input processing

By the time the message arrives for terminal input, it already has been edited by routines such as Basic Edit, ISC (Intersystem Communication), and MFS Edit.

The Front-End Switch exit routine gains control from an IMS system when the first segment of an input message is received before IMS determines the destination of the message. All input from FES-capable nodes and from ISC links are processed by this exit routine. Both MFS Edit and Basic Edit can remove characters that have a value less than X'41'.

For a diagram of the relationships among the front-end system, the intermediate system, and the back-end system with regard to message switching, see the following figure.

This exit routine can do any of the following:

- Indicate a destination change for an input message to an IBE destination or local transaction program defined in this IMS system. Changing the destination forces the originating terminal to be in response mode. (Front-end system processing.)
- Indicate a destination change for an input message from an IBE link to another IBE destination or to a local transaction program defined in this IMS system. (Intermediate system processing.)
- Define a transaction code that can be initiated when a specified time interval expires after switching the message. (Timeout processing.)
- Specify the message that can be sent directly to the input terminal for timeout processing.
- Provide IMS with additional routing information to expand the original message for any IBE system.
- Specify the name of a transaction program (full-function response mode or Fast Path) that processes or logs input messages due to user exit routine failures detected in other than the original terminal input (for example, ISC input).

The exit routine must provide additional routing information to identify the reply to this message when it comes back to the IMS front-end system. The user can tell IMS to remove the added information before the reply message is sent to the original terminal.

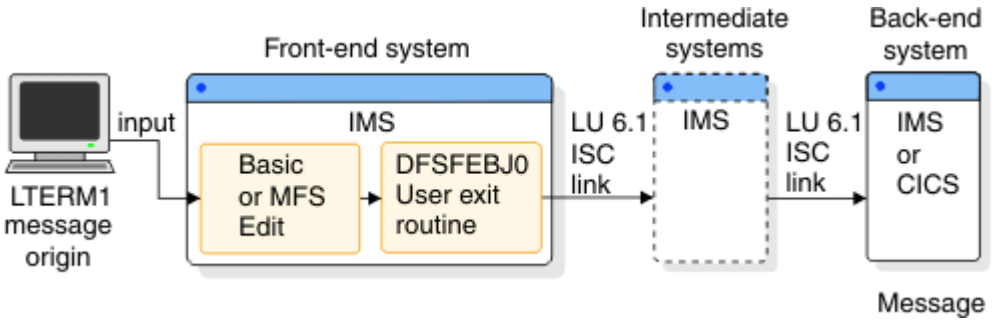


Figure 9. Message flow with the front-end switch exit routine

In the preceding figure, the reply path is not shown to keep the diagram simple. The reply would usually follow the same path back through the intermediate system or systems to the front-end, and then to the originating terminal.

IBE input processing

Correlate the reply message to a previously switched input message as part of IBE input processing. A reply to an input message, when received from another system, is treated by IMS as an input message.

The exit routine takes control of each message that comes from an ISC or FES-defined link. You must correlate the reply message to a previously switched input message.

The exit routine at this point can:

- Analyze the message text.
- Copy the LTERM name from the message text into the FEIBLTRM field.
- Copy the message identifier from the message text into the FEIBUNID field.
- Specify a destination for a late reply message in the FEIBLDST field.
- Tell IMS to remove the routing data from the message by specifying a length > 0 in the FEIBULNG field.
- Set the FEIBRPQ1 indicator if the reply message has to be sent directly to the original input terminal.
- Indicate the change of the destination code to a local transaction code (full-function non-response mode) in the FEIBNDST field.
- Set FEIBRPN to an error processing program (ERP) name that receives the input message if errors are detected in the verification of the exit parameters. An error message appears on the MTO of the system detecting the error.

Front-end interface block

A Front-End Interface Block (FEIB) is created for each FES capable terminal. The FEIB is used to communicate between the Front-End Switch exit routine and IMS.

For a VTAM node (excluding ISC) defined as FES capable (by an OPTIONS=FES on the TERMINAL, or TYPE macro, or ETO logon descriptor), the FEIB is allocated when the session has been established. The block is released when the VTAM session terminates and no reply for an FES message is outstanding.

Related Reading: For more information on the Extended Terminal Option (ETO) feature, see *IMS Version 15.2 Communications and Connections*.

The interface block is also allocated for each ISC parallel session. This is done automatically without special system definition at LOGON or OPEN DEST time. The interface block is destroyed at LOGOFF time, at CLOSE DEST time, or at session failure.

If the exit routine is not defined in the system or if the VTAM node is not defined as FES capable, the FEIB will not be allocated.

Register 1 on entry to the exit contains the address of the interface block.

The FEIB layout is in the following example.

```
*-------------------------------------------------------------------*
*      FEIB - FRONT END MESSAGE SWITCH INTERFACE BLOCK DSECT         *
*-------------------------------------------------------------------*

FEIB     DSECT
FEIBIFLG DS    X                   USER EXIT INPUT FLAGS
FEIBISC  EQU   X'80'               MESSAGE FROM AN ISC LINK
*        EQU   X'40'               RESERVED BY IBM
*        EQU   X'20'               RESERVED BY IBM
*        EQU   X'10'               RESERVED BY IBM
*        EQU   X'08'               RESERVED BY IBM
*        EQU   X'04'               RESERVED BY IBM
*        EQU   X'02'               RESERVED BY IBM
*        EQU   X'01'               RESERVED BY IBM
FEIBOFLG DS    X                   USER EXIT OUTPUT FLAGS
FEIBRPQ1 EQU   X'80'               QUEUE RESPONSE TO ORIG DEVICE
*                                  ELSE QUEUE SMB NAMED IN FEIBNDST
FEIBERP  EQU   X'40'               ON TIMEOUT CALL ERP, ELSE ERR MSG
*        EQU   X'20'               RESERVED BY IBM
FEIBTMED EQU   X'10'               TIME RESPONSE WITH SYSDEF VALUE
*        EQU   X'08'               RESERVED BY IBM
*        EQU   X'04'               RESERVED BY IBM
*        EQU   X'02'               RESERVED BY IBM
*        EQU   X'01'               RESERVED BY IBM
FEIBMSGN DS    H                   TIMEOUT ERROR MESSAGE NUMBER
*                                  ONLY USED IF FEIBERP OFF
FEIBLTRM DS    CL8                 LTERM NAME OF ORIGINAL TERMINAL
*                                  ONLY AVAILABLE IF FEIBISC OFF
FEIBMSG  DS    A                   POINTER TO INPUT MESSAGE BUFFER
```

```
FEIBUNID DS    F                   UNIQUE ID NUMBER (FULL WORD BIN)
FEIBNDST DS    CL8                 NAME OF NEW DEST TO QUEUE MESSAGE
FEIBERPN DS    CL8                 NAME OF ERP TO CALL ON TIMEOUT
*                                  ONLY USED IF FEIBERP ON
FEIBLDST DS    CL8                 NAME OF DEST TO QUEUE LATE MESSAGE
FEIBULNG DS    H                   LENGTH OF DATA IN USER AREA
FEIBUSER DS    CL40                USER AREA FOR DATA TO PREFIX MSG
*                                  ONLY USED IF FEIBULNG > 0.
FEIBIMID DS    CL4                 IMS IDENTIFIER
FEIBTIME DS    H                   TIMEOUT INTERVAL (SECONDS)
FEIBPRN DS     CL8                 PRIMARY RESOURCE NAME ADDED
                                   TO USER DATA BY ISC EDIT
```

Description of the FEIB fields

Correlate the reply message to a previously switched input message by associating message components with the values in the front-end interface block (FEIB) fields.

The following table provides a description of the FEIB fields.

Table 51. Description of the FEIB fields

Field	Description
FEIBIFLG	Input flag: **FEIBISC (bit 0)** • on: message is from an ISC link • off: message is from an FES capable device **FEIBISC (bits 1–7)** Reserved
FEIBOFLG	Output flags: **FEIBRPQ1 (bit 0)** • on: reply message has to be sent directly to the original input terminal • off: reply message has to an SMB named in FEIBNDST **FEIBERP (bit 1)** • on: on timeout, schedule the SMB named in FEIBERPN • off: on timeout send text of error message defined in FEIBMSGN to the original input terminal (only used if FEIBTMED is ON.) **FEIBTMED (bit 3)** • on: release terminal from response mode when the timeout value is exceeded • off: timeout facility is not used for this message **FEIBDELT (bit 4)** • on: defer timeout facility until FP sync–point • off: timeout facility will be activated immediately at input message processing (only used if FEIBTMED is ON) **FEIBDELT (bits 2, 5–7)** Reserved
FEIBMSGN	User message number from table (DFSCMTU0) which is sent to the original input terminal in the case of a timeout. The message number can only be specified if the FEIBERP bit is off. Values range from 1-999. (Binary Number.)
FEIBLTRM	Logical Terminal Name (LTERM) of the input terminal. For a reply message, DFSFEBJ0 must store the LTERM name into this field, padding with blanks on the right.

Table 51. Description of the FEIB fields (continued)

Field	Description
FEIBMSG	Pointer to the DC buffer containing the input message.
FEIBUNID	Unique message identifier is only available if the FEIBISC bit is off on input to the exit routine. The exit routine must store the unique identifier (a binary number) into this field.
FEIBNDST	New destination name for the message. This identify an IBE destination or a transaction code. (Blank padded on right.)
FEIBERPN	Error processing program name (transaction code) to be scheduled in the case of a timeout. The FEIBERP bit must be set on if the program name is specified. (Blank padded on right.) This field is also used to specify an optional ERP if the input is from an IBE session. In this case, FEIBERP need not be set, and the ERP is scheduled with the input from the IBE session.
FEIBLDST	Transaction name that is scheduled when a reply message arrives after timeout. (Blank padded on right.)
FEIBULNG	This field must contain the length of the user data for an input message. It is used by IMS to expand the original message. This field can contain the length of the user data to be removed by IMS from the reply message for an output message.
FEIBUSER	User data area for routing information that IMS uses to expand the message. This field is used for input messages only if the FEIBULNG field is greater than zero.
FEIBIMID	At input to the exit routine, this field contains the identifier for the IMS system as specified during system definition on the IMSCTRL macro.
FEIBTIME	Timeout interval override (in seconds). This field is used to override the system Front-End-Switch timeout value as supplied in IMS startup parameters. If a value of 0 is in this field, the system default override value is used.
FEIBPRN	At input to the exit routine, this field contains the primary resource name that was added to the user data by ISC edit.

The following table shows the FEIB usage.

Table 52. FEIB usage

Entry name and data Type	Input message processing						Reply message processing					
	Front-end system		Back-end system		Intermediate system		Front-end system		Intermediate system		Back-end system	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
FEIBMSG DS A	X		X		X		N/A	N/A	X		X	
FEIBLTRM DS CL8	X		X		X		N/A	N/A	X		X	X
FEIBERPN DS CL8		X		X			N/A	N/A		X		X
FEIBMSGN DS H		X					N/A	N/A				
FEIBNDST DS CL8		X		X			N/A	N/A		X		X
FEIBUNID DS F	X						N/A	N/A				X

Table 52. FEIB usage (continued)

Entry name and data Type	Input message processing						Reply message processing					
	Front-end system		Back-end system		Intermediate system		Front-end system		Intermediate system		Back-end system	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
FEIBLDST DS CL8							N/A	N/A				X
FEIBULNG DS H		X					N/A	N/A				X
FEIBUSER DS CL40		X					N/A	N/A				
FEIBISC EQU BIT	X(0)		X(1)		X(1)		N/A	N/A	X(1)		X(1)	
FEIBRPQ1 EQU BIT							N/A	N/A				X
FEIBERP EQU BIT		X					N/A	N/A				
FEIBTMED EQU BIT		X					N/A	N/A				
FEIBIMID DS CL4	X		X		X		N/A	N/A	X		X	
FEIBTIME DS H		X					N/A	N/A				
FEIBPRN DS CL8	X		X		X		N/A	N/A				
Return code (R15)	2-New destination from FE		6-New destination from IBE		0-Nothing		N/A		6-New destination from IBE		8-Reply	

Note: X(0) = off X(1) = on

Related reference

“Routing information” on page 169
You are responsible for the format and the contents of the routing information.

Input and output fields

Depending on the system, front-end interface block (FEIB) fields will be used for input, which are stored by IMS, while other FEIB fields will be used for output and stored by the Front-End Switch exit routine.

The following table show the input fields and the output fields

Table 53. FES data flow for input message processing

System	Input	Output
Front-end system	FEIBLTRM (CL8)	FEIBERPM (CL8)
	FEIBMSG (A)	FEIBMSGN (H)
	FEIBUNID (F)	FEIBNDST (CL8)
	FEIBIMID (CL4)	FEIBUSER (CL40)
	FEIBPRN (CL8)	FEIBULNG (H)
		FEIBTIME (H)
		Return codes:
		0 - nothing
		12 - Table error
	Flags: FEIBISC	Flags: FEIBERP, FEIBTMED
Intermediate system	FEIBMSG (A)	FEIBNDST (CL8)
	FEIBIMID (CL4)	FEIBERPN (CL8)
	FEIBLTRM (CL8)	
	FEIBPRN (CL8)	
		Return codes:
		6 - New destination from IBE
		12 - Table error
	Flags: FEIBISC	Flags: N/A
Back-end system	FEIBMSG (A)	
	FEIBIMID (CL4)	
	FEIBLTRM (CL8)	
	FEIBPRN (CL8)	
		Return codes:
		0 - Nothing
	Flags: FEIBISC	Flags: N/A

The following table shows the input and output fields for reply message processing.

Table 54. FES data flow for reply message processing

System	Input	Output
Front-end system	FEIBMSG (A)	FEIBLTRM (CL8)
	FEIBIMID (CL4)	FEIBUNID (F)
	FEIBLTRM (CL8)	FEIBLDST (CL8)
		FEIBNDST (CL8)
		FEIBULNG (H)
		FEIBERPN (CL8)
		Return codes:
		0 - nothing
		8 - Reply
		12 - Table error
	Flags: FEIBISC	Flags: FEIBRPQ1
Intermediate system	FEIBMSG (A)	FEIBNDST (CL8)
	FEIBIMID (CL4)	FEIBERPN (CL8)
	FEIBLTRM (CL8)	
		Return codes:
		0 - Nothing
		6 - New destination from IBE
		12 - Table error
	Flags: FEIBISC	Flags: N/A
Back-end system	N/A	N/A

Routing information

You are responsible for the format and the contents of the routing information.

If the value of the FEIBULNG field is greater than zero, IMS adds the user data on an input message from an FE device to the input message between the old destination and the message text.Both MFS edit and Basic Edit can remove characters that have a value less than X'41'. As part of the routing information, the following is required:

- A unique identifier assigned to the input message from the originating terminal. This identifier must be sent with the user data to identify the reply to this message when it comes back to IMS. For messages being processed by either MFS or Basic Edit, the identifier value must be translated into unpacked format.
- The LTERM name of the originating terminal. IMS does not have access to the control blocks of the originating terminal when the reply to a switched message arrives. Therefore the exit routine must add the LTERM name of the originating terminal to the user data. This LTERM name is to be rerouted with the reply from the back-end system and must not be removed or changed by any intermediate system.

When the exit routine gains control from IMS on input of the reply message, it obtains the LTERM name and the unique identifier from the message text and stores them into the corresponding fields of the FEIB. IMS then determines the original input terminal and checks if timeout has already occurred. The destination of the message is determined by the result of this check.

If the timer has not expired, one of the following occurs:

- The message is sent directly to the original input terminal.
- The message is queued to a local transaction, which can cause a reply message to be sent to the originating LTERM using the I/O PCB.

Be aware that the TPCBTSYM field of the I/O (TPPCB) might contain the ISC LTERM name when the application does an ISRT reply back to the originating LTERM. This choice is decided by the exit routine.

If the timer has expired, the message is no longer expected at the original terminal, because it is already released from response mode. The message is then sent to the destination defined by the exit routine for late reply messages.

Besides required routing information, the routine can store additional information, such as a unique system identification throughout all connected systems.

Application programs processing FES messages must understand that the input message contains routing information which must be rerouted to the front-end system. The routing information in all the involved systems must be in agreement. The routing information in the input message must be included in the output message.

Message expansion

Combine the original message with the routine information and store it in the new buffer.

Because the DC buffer is not large enough to store the routing information, use the FEIBUSER field of the FEIB. The length of the user data must be stored in the FEIBULNG field of the FEIB. The maximum length of user data is 40 bytes. IMS combines the original message with the user data and stores both into the new buffer. The new destination (FEIBNDST) is also stored into the new buffer.

The following figure shows the original and new buffer formats.

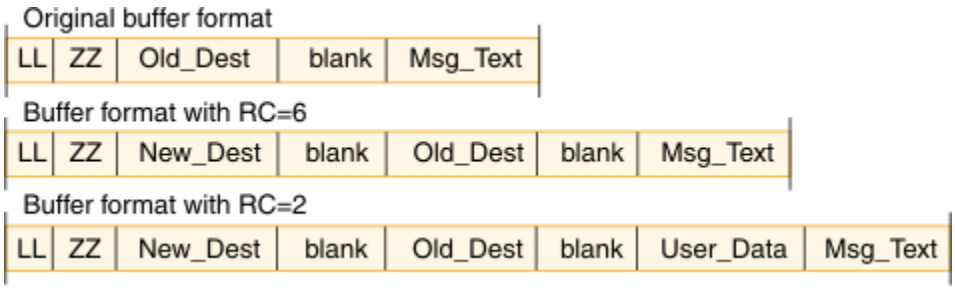


Figure 10. Old and new buffer formats

New_Dest
New destination from FEIBNDST field

User_Data
User data from FEIBUSER field

The old destination and the new destination are both followed by a blank. You must lay out the routing information. After IMS has expanded the message, the routing information should precede the original message text.

Timer facility

The timer facility controls each input message that is routine to a back-end system.

When the specified time interval expires without a reply to the input message, the input terminal is released from response mode. The timeout value is specified by the FESTIM parameter on the IMS procedure, and can be overwritten by specifying a non-zero value in the FEIBTIME field during front-end

processing of an input message. To make use of the timer, set the FEIBTMED flag in the FEIB. In addition, you must specify the action which has to be taken at timeout. This can be done by specifying either the name of a program that is to be given control (FEIBERPN field) or a message that is to be issued (FEIBMSGN field). The message number must be included in the user message table DFSCMTU0. See DFSCMTU0 for more information. The program can send a message to the input terminal using the I/O PCB. This response releases the terminal from response mode. The message text is directly sent to the input terminal if you define a message number.

If the reply comes in time, the timer request for the input message is canceled. No timeout can occur if you do not set the FEIBTMED indicator. If no reply is received, the terminal is not released from response mode.

If the input terminal is in an active conversation status, the timer facility will not be activated.

When switching to a local Fast Path transaction, the timeout facility can be deferred until Fast Path sync-point by setting the FEIBDELT flag.

FEIBRPQ1 indicator

The FEIBRPQ1 indicator must be set in the FEIB for a reply message to be sent directly to the original input terminal.

This indicator can only be set when a reply message has a return code of 8 in register 15. If you do not set it, you have to store a new destination into the FEIBNDST field of the FEIB. IMS checks the indicator and sends the message, depending on the values in the FEIB.

If you change the destination code of an input message to a local transaction which sends a message across a link, the timer supervisor includes the elapsed time for the local transaction.

If the destination of a reply message is changed to a local transaction, the original input terminal is released from the timer supervisor before the local transaction is scheduled. If the transaction is not available or if the application program does not send an output message to the original input terminal, the terminal is not released from response mode.

Example of the front-end switch exit routine (DFSFEBJ0)

A front-end switch exit routine allows you to keep the front-end system in response mode while it is waiting for the reply from the intermediate system for messages entered in a back-end system.

Subsections:

Routing scheme

In the following figure, three IMS systems are connected by ISC links. SFIMS2 acts as the front-end system, and LAIMS1 and NYIMS1 can act as a back-end system. In addition, LAIMS1 can act as an intermediate system.

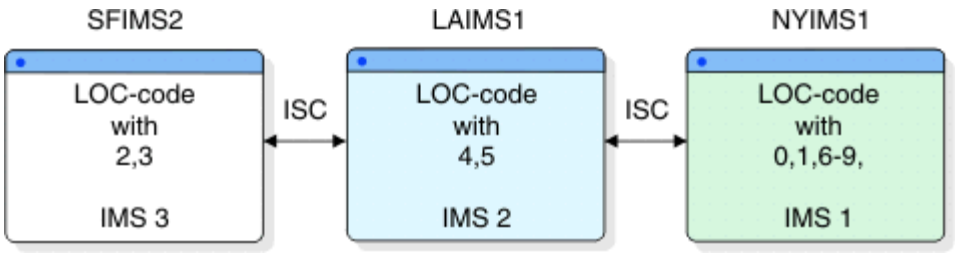


Figure 11. Routing scheme of front-end switch exit routine example

In each system, you can enter a transaction FESTX1. This is not defined as a transaction in the system, but is a special transaction code used by the sample exit routine that identifies this message as an FES

transaction. The exit routine in the front-end system (SFIMS2) changes the transaction code to FESTX2, which must be defined in the system as a valid transaction.

There is an eight-digit location code (LOC-code) in the user data. The decision as to which system processes the transaction depends on this LOC-code. If the transaction is to be processed in another system, the exit routine changes the destination to LAIMS1 so that either LAIMS1 or NYIMS1 processes the transaction FESTX2.

The following location codes are defined:

System	Location code (LOC-code)
SFIMS2	20000000 - 39999999
LAIMS1	40000000 - 59999999
NYIMS1	00000000 - 19999999 60000000 - 99999999

The system that processes the transaction FESTX2 generates an output message containing the transaction code FESTX3 in front of the message text. As with FESTX1, this is not defined as a transaction in the system, but is a special transaction code used by the sample exit routine that identifies this message as a reply to an FES transaction. This output message has to be routed to the front-end system where the corresponding FESTX1 transaction was entered which is now the target system for the reply message.

The following tables show routing information for each system.

Table 55. SFIMS2 tables

SFIMS2 - table I		**SFIMS2 - table II**[1]	
1st digit of LOC-code	Next system	Target system	Next system
0	LAIMS1	LAIMS1	LAIMS1
1	LAIMS1	NYIMS1	LAIMS1
4	LAIMS1		
5	LAIMS1		
6	LAIMS1		
7	LAIMS1		
8	LAIMS1		
9	LAIMS1		

Note: [1] This table is used only if it is an intermediate system

Table 56. LAIMS1 tables

LAIMS1 - table I		**LAIMS1 - table II**	
1st digit of LOC-code	Next system	Target system	Next system

Table 56. LAIMS1 tables (continued)			
LAIMS1 - table I		**LAIMS1 - table II**	
0 1 2 3 6 7 8 9	NYIMS1 NYIMS1 SFIMS2 SFIMS2 NYIMS1 NYIMS1 NYIMS1 NYIMS1	SFIMS2 NYIMS1	SFIMS2 NYIMS1

Table 57. NYIMS1 tables			
NYIMS1 - table I		**NYIMS1 - table II[1]**	
1st digit of LOC-code	Next system	Target system	Next system
2 3 4 5	LAIMS1 LAIMS1 LAIMS1 LAIMS1	LAIMS1 SFIMS2	LAIMS1 LAIMS1

Note: [1] This table is used only if it is an intermediate system

Description of sample exit routine

The example in this section is based on the assumption that ISCEDIT is used for editing the messages going across ISC links. ISCEDIT removes the first data field of the message text on output to an ISC destination.

The exit routine is designed to run in each of the three systems without modifying the code. It has to process different tables with routing information for each system, and has to know the name of the owning system. This is obtained from the FEIBIMID field. In this example:

- NYIMS1='IMS1' back-end system
- LAIMS1='IMS2' back-end or intermediate system
- SFIMS2='IMS3' front-end system

The exit routine in each system must analyze the transaction code and the LOC-code in the message text:

- If the transaction code is FESTX1, and
  - Change the transaction code to FESTX2.
  - If the LOC-code is in table I:
    - Change the transaction code to FESTX2.
    - Change the destination to the corresponding destination from table I (FEIBNDST).
    - Set the FEIBTMED indicator on, if appropriate.
    - Set the FEIBERP indicator on, if appropriate.
    - Set the transaction code for ERP (FEIBERPN), if appropriate.
    - Store the following routing information into the user area of the FEIB (FEIBUSER) as shown in the following figure.

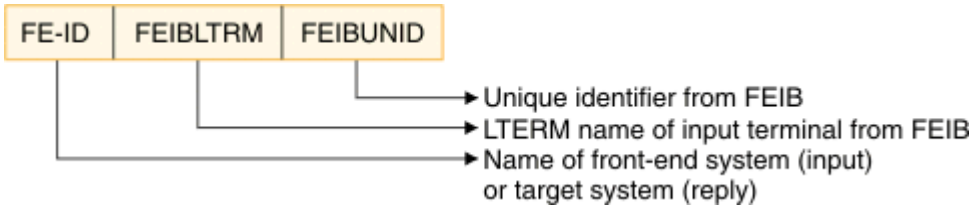


Figure 12. User area of FEIB (FEIBUSER)

The FEIBUNID value is unpacked into zoned format to prevent MFS Edit or Basic Edit from removing characters less than X'41'.

- Set the user data length field to 24 (FIEBULNG).
- Set the RC=02 in register 15.

– Else Set RC=00 in register 15.

- If the transaction code is FESTX2 and the LOC–code is in table I:
  – Change the destination to the corresponding destination from table I (FEIBNDST).
  – Set the RC=06 in register 15.
- If the transaction code is FESTX3:
  – Analyze the routing information.
  – If the name of the target system in the routing information (FE–ID) is not the name of the owning system:
    - Change the destination to the corresponding destination from table II (FEIBNDST).
    - Set the RC=06 in register 15.
  – If the name of the target system is not table II, set RC=12 in register 15.
  – If the name of the target system in the routing information is the name of the owning:
    - Get the LTERM name from the routing information and store it into the interface block (FEIBLTRM).
    - Get a unique identifier from the routing information, change it from zoned to packed format, and store it in the interface block (FEIBUNID).
    - Set the transaction code for a message which comes too late (FEIBLDST).
    - Set the FEIBRPQ1–indicator.
    - Set the user data length field to 31 (FIEBULNG).
    - Set the RC=08 in register 15.
- In all other cases no action is taken by the exit routine.[1]

Global Physical Terminal (Input) edit routine (DFSGPIX0)

The Global Physical Terminal (Input) edit routine (DFSGPIX0) is called before the IMS Basic Edit routine and eliminates the overhead associated with defining the edit routine for each terminal through system definition.

This topic contains Product-sensitive Programming Interface information.

This topic describes the Global Physical Terminal Input edit routine. This routine is a user-written edit routine that performs the same functions as the Physical Terminal Input edit routine (DFSPIXT0).

Subsections:

- "About this routine" on page 175
- "Communicating with IMS" on page 177

[1] In an IMS back–end system, which processes TX2, an application program generates the output message with TX3.

About this routine

If you write and include the routine in your system, IMS calls it for all terminals that do not have the Physical Terminal Input edit routine specified. By using the Global Physical Terminal Input edit routine instead of the Physical Terminal Input edit routine, you can eliminate the overhead associated with defining the edit routine for each terminal through system definition.

If the input message is processed by MFS, the Global Physical Terminal (Input) edit routine is not called. This edit routine is only called when a non-LU 6.2 message is entered from a terminal; it is not called when the message is inserted by a program-to-program switch.

Message segments are passed one at a time to the Global Physical Terminal (Input) edit routine, and the edit routine can handle them in one of the following ways:

- Accept the segment and release it for further editing by the IMS Basic Edit routine.
- Modify the segment (for example, change the transaction code or reformat the message text) and release it for further editing by the IMS Basic edit routine. Examples of segment modifications that can be made are:
  - changing the transaction code.
  - reformatting the message text.
- You can make any required modifications within the original segment because IMS has not yet performed destination or security checking.
- You cannot alter the length of this segment.
- Cancel the segment.
- Cancel the message and request that IMS send a corresponding message to the terminal operator.
- Cancel the message and request that IMS send a specific message from the User Message Table to the terminal operator.

The following table shows the attributes of the Global Physical Terminal (Input) Edit exit routine.

Table 58. Global physical terminal input edit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSGPIX0.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS callable services with this routine, you must do the following: • Issue initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found in register 9 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.

Table 58. Global physical terminal input edit routine attributes (continued)

Attribute	Description
Sample routine location	No sample exit routine is provided. Instead, use the IMS.ADFSSMPL distribution library (member name DFSPIXT0). The sample is identical to the Physical Terminal (Input) edit routine (DFSPIXT0), because the two edit routines perform the same function. This routine performs the following functions: • Scans the input message segment for an expected format—TESTEXIT. • Generates return codes (XX) based on the input request (TESTEXIT,XX). • Verifies the user message number (YYY) if specified (TESTEXIT,XX,YYY). • Replaces TESTEXIT with ERROR if return code or message number is invalid and passes the segment to IMS (return code 0). **Note:** The sample exit routine is not reentrant. You must assemble it with PARM='OBJECT,NODECK,NORENT' and link-edit it with PARM='NCAL,LET,LIST,XREF,SIZE(880K,64k)'.

Bypassing Basic Edit

If the IMS application program supplies DFS.EDTN in the MOD name parameter for the output message, IMS bypasses the Basic Edit routine, except for transaction code and password validation.

Related Reading: For further information see "MFS Bypass for 3270 or SLU 2" in the "Application Programming Using MFS" in *IMS Version 15.2 Application Programming APIs*.

The Physical Terminal Input edit routine must position the transaction code, and optionally the password, if the terminal is not operating in conversational or preset destination mode. The edit routine should detect errors and have IMS send a message to the terminal operator if the routine finds any errors.

IMS maintains a flag in the CTB (bit CTB6TRNI in the CTBFLAG6 field) to indicate when 3270 MFS bypass, nonconversational, no preset destination and first segment exist on input to the Global Physical Terminal (Input) edit routine. This flag notifies the Global Physical Terminal Input edit routine that it can add a minimum of one byte and a maximum of 18 bytes to the front of the message segment for a transaction code and optional password. The minimum of one byte to be added to the front of the message segment consists of a one-byte transaction code. If NOBLANK is not specified at system definition, a minimum of two bytes is added to the front of the message segment, consisting of a one-byte transaction code and one blank, which is necessary as a separator. To add a transaction code and optional password, the exit routine can put a return code of 16 in register 15 and set register 1 to point to an LLZZ field followed by the data to be added. You cannot, however, alter the length of the segment passed in to the exit. If you need to insert a transaction or destination code, and an optional password, set register 1 to the address of a static data field that consists of a halfword length (LL), a halfword of binary zeroes (ZZ), and zero to 14 bytes of user data.

Specifying the routine

You must assemble and bind the edit routine into the IMS execution time library or user library concatenated in front of the IMS execution time library.

IMS calls the Global Physical Terminal Input edit routine (DFSGPIX0) for each terminal that does **not** have EDIT=(,YES) coded on the TERMINAL macro or ETO logon descriptor.

For terminals that do have EDIT=(,YES) specified on the TERMINAL macro or ETO logon descriptor, IMS calls the Physical Terminal Input edit routine (DFSPIXT0).

Related Reading:

- For more information on the TERMINAL macro, see *IMS Version 15.2 System Definition*.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Content
1	Address of the input message segment buffer. IMS editing has not been performed. The first two bytes of the buffer contain the segment length (binary length includes the 4-byte overhead). The third and fourth bytes of the buffer are binary zeros. The message text begins in the fifth byte of the buffer. If the device was defined with MFS support, but this message is not being processed by MFS, the first segment of the message has backspace error correction performed before entry to this edit routine. If escape (**) was entered by the terminal operator, the first two data bytes have been changed to binary zeros.
7	Address of CTB for the physical terminal from which the message was entered.
9	Address of CLB for the physical terminal from which the message was entered.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

The edit routine you supply can edit the message segment in the buffer pointed to by register 1.

You can reduce the length of the message segment to any size by replacing the length in the buffer with the appropriate value. The length field must appear in the same place at exit as at entry, and bytes 3 and 4 must not be changed.

Contents of registers on exit

Before returning to IMS, the edit routine must restore all registers except for register 1, which contains a message number if register 15 contains a value of 12; otherwise register 1 is ignored. Register 15 contains one of the following return codes:

Return codes	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled, and the message identified by register 1 is sent to the terminal.
16	Insert the transaction code and optional password following the LLZZ pointed to by register 1. This return code is only valid for 3270 MFS bypass terminals. When the entering terminal is not a 3270 MFS bypass terminal, and the physical terminal input exit gives a return code of 16, IMS issues an error message, and the transaction code is not inserted in the message.

Any other return code causes the message to be canceled and the terminal operator to be notified.

Related reference

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"Physical Terminal (Input) edit routine (DFSPIXT0)" on page 260
The Physical Terminal (Input) edit routine (DFSPIXT0) user-written edit routine gains control before the IMS Basic Edit routine. It is used to accept, modify, and cancel segments and messages.

Greeting Messages exit routine (DFSGMSG0)

The Greeting Messages exit routine (DFSGMSG0) allows you to tailor how IMS handles messages issued during the logon and signon process.

The exit also allows you to:

- Change the MFS Message Output Description (MOD) name without changing the message. (However, if the terminal is the Master Terminal and is master formatted, the request to change the MOD name is ignored.)
- Change the message without changing the MOD name.
- Send a null message (no data) for formatting purposes.
- Display or process RACF messages that are issued at signon.

Subsections:

About this routine

IMS builds a message based on the calling module's request. This message, plus information useful to the exit and a buffer for returning an alternate message built by the exit, are passed as input to the exit. The exit indicates by a return code if the message built by IMS should be used, or if an alternate message has been returned and should be used. The message length returned must be at least five bytes (four bytes for the length field and a one-byte message).

The following table shows the attributes of the Greeting Messages exit routine.

Table 59. Greeting messages exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSGMSG0.
Including the routine	You can assemble the sample exit routine, or one that you write (using the standard IMS macro and copy files), and include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. If the Greeting Messages exit routine is included, IMS automatically loads it each time IMS is initialized.
IMS callable services	To use IMS callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found at offset 0 of the Greeting Messages Exit parameter list. • Link DFSCSI00 with your user exit.

Table 59. Greeting messages exit routine attributes (continued)

Attribute	Description
Sample routine location	IMS.ADFSSMPL (member name DFSGMSG0). The sample exit uses the DFS3649 and DFS2467 messages built by IMS, but it converts the DFS3650 message to a single-segment message. You can also write your own exit routine.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
13	Save area address
14	Return address to IMS
15	Entry point address of exit routine

The following table shows the greeting messages exit parameters. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 60. Greeting messages exit parameter list

Offset	Length	Description
+0	4	ECB address.
+4	4	SCD address.
+8	4	Pointer to User Table.
+12	4	Address of parameter list for this exit. For additional details on the content and the format of these parameters, see the prolog in the sample routine.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains the return code. The returns codes are as follows:

Register	Contents
15	One of the following return codes: **Return code** **Meaning** **X'00'** Use the message built by IMS. **X'04'** Use the message in the alternate buffer (single segment). **X'08'** Use the message in the alternate buffer (multiple segment). **X'0C'** Send a null message so that the device is formatted with the MFS format specified by IMS or returned by the exit. **X'10'** Bypass password verification. Valid only for message DFS3656A.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

Related information

DFS3656A (Messages and Codes)

IMS Adapter for REXX exit routine (DFSREXXU)

The IMS Adapter for REXX exit routine (DFSREXXU) gets control before the environment is built, just before an exec is executed, and just after it ends.

You can use DFSREXXU with the IMS adapter for the REXX environment. It is optional and can be omitted from the bind step. The user exit routine is used more for an installation than for a specific execution. The user exit routine is provided by the IMS adapter for REXX and is called only when a new REXX transaction is scheduled and ends. The user exit is not associated with the standard REXX exits provided by TSO. A sample user exit routine (DFSREXXU) is shipped with IMS (in source code only). For the latest version of the DFSREXXU source code, see the IMS.SDFSSRC distribution library; member name is DFSREXXU.

The routine has the ability to do the following:

- Override the exec name to be executed. This name defaults to the IMS program name.
- Choose not to execute any exec and have the IMS adapter for REXX return to IMS.

  It is the exit routine's responsibility to do any required processing such as issuing a GU (Get-Unique) call if the region type is MPP.
- Issue DL/I calls using the AIB interface as part of its logic in determining what exec to execute.
- Set REXX variables (through IRXEXCOM) before the exec is started. The variables are then available to the exec.
- Extract REXX variables (through IRXEXCOM) after the exec ends. These variables were set earlier by the exec or exit routine.
- Change the initial default IMSRXTRC tracing level.

The user exit routine must conform to all of the restrictions that apply to IMS application programs.

Subsections

About this routine

The following table shows the attributes of the IMS Adapter for REXX exit routine.

Table 61. IMS Adapter for REXX exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL.
Naming convention	The user exit routine must be named DFSREXX0.
Binding	You must bind the user exit with DFSREXX1 during installation of the IMS adapter for REXX.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSRC distribution library.

The routine must be written to be reentrant (RENT), AMODE 31, RMODE ANY.

Parameters

Entry parameters are:

R0
Pointer to REXX Environment Block as described in *z/OS TSO/E REXX Reference*.

R1
Pointer to parameter list

R13
Pointer to save area

R14
Return address

R15
Entry point address

On exit, all registers except R15 must be restored. Only the parameters can be altered. The content of R15 is ignored on exit.

The parameter list contains a list of pointers to the parameters. All character data is left justified and padded with blanks, if necessary. Omitted fields are set to blanks. All fields are read-only unless otherwise specified. The following table shows the user exit parameter list format.

Table 62. User exit parameter list

Name	Offset (decimal)	Data type	Length (decimal)	Description
Function	0	Pointer	4	Pointer to one word function type. Func=0 on Setup Call, Func=1 on Entry Call, Func=2 on Exit Call.

Table 62. User exit parameter list (continued)

Name	Offset (decimal)	Data type	Length (decimal)	Description
EXECParm	4	Pointer	4	Pointer to 128-byte area containing parameters that are passed to the REXX interpreter. The format of the area is a halfword length field that contains the length of the text string that follows. The first blank separated word or the entire string if no blanks are present is the exec name to execute. On entry this field is set to the program name followed by one blank and the transaction code if available. The exit can rebuild this field when called on entry to alter the exec name or parameters that are passed. The length field can be set to zero indicating no exec is to be executed.
PgmName	8	Pointer	4	Pointer to 8-byte area containing the Program name that was scheduled.
TranCode	12	Pointer	4	Pointer to 8-byte area containing the Transaction Code that was scheduled, if available (MPP,BMP,IFP).
User_ID	16	Pointer	4	Pointer to 8-byte area containing the current user ID for the scheduled program, if available (MPP,BMP,IFP).
IMSRXTRC	20	Pointer	4	Pointer to one word IMSRXTRC level. This value defaults to 1 at exec startup but can be overridden by the user exit. See *IMS Version 15.2 Application Programming* for more information on values. Note that the level field here is a FULLWORD and not EBCDIC.
UserArea	24	Pointer	4	Pointer to 8-byte (word aligned) user area that is passed on entry and is preserved verbatim on exit. This field is set to binary zeros whenever the REXX environment is built in the dependent region. The user area can be altered by the user exit and is provided as an anchor.
RetCode	28	Pointer	4	Pointer to one word return code. The return code must be set to zero.
UseridInd	32	Pointer	4	Pointer to one-byte User ID Indicator that describes the content of the user ID field. The indicator can be: U-User ID, L-LTERM, P-PSBname, or O-Other.

Note:

1. When on a Setup call the next four parameters are not available; their addresses are 0.

For each user exit parameter described in the preceding table, the following table shows the corresponding DFSREXXU parameter.

Table 63. DFSREXXU parameter list

User exit parameter	DFSREXXU parameter
Function pointer	FUNCTION_CODE DS F FUNC_SETUP EQU 0 FUNC_BEFORE EQU 1 FUNC_AFTER EQU 2
EXECParm pointer	EXEC_PARM DS 0CL128 EXEC_PARM_LL DS H EXEC_PARM_TXT DS CL126
PgmName pointer	PGM_NAME DS CL8

Table 63. DFSREXXU parameter list (continued)

User exit parameter	DFSREXXU parameter
TranCode pointer	TRAN_CODE DS CL8
User_ID pointer	USER_ID DS CL8
IMSRXTRC pointer	IMSRXTRC_LEV DS F
UserArea pointer	USER_AREA DS 2F
RetCode pointer	RETURN_CODE DS F
Useridind pointer	USERID_IND DS F

Related concepts

z/OS: Using the environment block

Initialization exit routine (DFSINTX0)

Use the Initialization exit routine (DFSINTX0) to create two user data areas that can be used by some of your installation's exit routines.

This topic contains Product-sensitive Programming Interface information.

IMS calls the Initialization exit routine during initialization as a common Transaction Manager exit routine. Certain IMS user exit routines are called before the DFSINTX0 user exit routine is called. These user exit routines are: DFSPSE00, DFSHINT0, DFSZINT0, RASE, any exit routine of type AOIE, and DFSQSPC0/DFSSSSP0.

- General user data area

  The address of this user data area is passed as part of the IMS standard user exit interface. Any exit routine that uses this interface will have access to this data area (if it exists). The address of this data area is also passed as part of the nonstandard interface to the following exit routines:

  Command Authorization exit routine (DFSCCMD0)
  Greeting Messages exit routine (DFSGMSG0)
  Logoff exit routine (DFSLGFX0)
  Logon exit routine (DFSLGNX0)
  Destination Creation exit routine (DFSINSX0)
  Signoff exit routine (DFSSGFX0)
  Signon exit routine (DFSSGNX0)

  The general user data area is not available to some IMS user exit routines when they are called during IMS initialization because the DFSINTX0 user exit routine is called during IMS initialization after these user exit routines are called. The user data area is not available to the following exit routines when they are called during IMS initialization: DFSPSE00, DFSHINT0, DFSZINT0, any exit routine of type AOIE, RASE, and DFSQSPC0/DFSSSSP0.

  Other TM exit routines can address the user data areas through SCDINTXP. See the topic for each exit routine for information on the routine's parameter list.

- LU 6.2 user data area

  The LU 6.2 user data area is not passed as part of the IMS standard user exit interface. It is passed as part the nonstandard interface to the LU 6.2 Edit exit routine.

You can also use this exit routine to alter the setting for the Extended Terminal Option (ETO) feature. You can leave ETO activated or override the setting to indicate that ETO is not required, even if you previously requested it.

This exit is also used to enable password verification. The IMS default processing is to disable password verification. With password verification, users signing on to VTAM terminals that change their password are prompted to verify the new password.

Subsections:

About this routine

The Initialization exit routine is optional. If the exit is included in the system, IMS calls it before IMS loads the ETO descriptors and any exit routine that requires ETO to be active. If ETO is required for an exit routine, the documentation for the routine states that requirement. If the Initialization exit routine returns a return code indicating that ETO should not be made available, the ETO exit routines and descriptors will not be loaded. If this exit is not included in the system, IMS proceeds using the setting for the ETO= keyword that is specified as an EXEC parameter or in the DFSPBxx of IMS.PROCLIB.

The initialization exit routine can optionally enable password verification and an alternate ETO ALOT=0 option by setting the appropriate flags in the exit routine parameter list.

Attributes of the routine

The following table shows the attributes of the Initialization exit routine.

Table 64. Initialization exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSINTX0.
Binding	This exit routine must be reentrant.
Including the routine	If you want IMS to call the Initialization exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. If the exit routine is included, IMS automatically loads it and calls it at initialization.
IMS callable services	DFSINTX0 can use callable storage services. To use IMS callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found at offset X'0' of the IMS Initialization exit parameter list. • Link DFSCSI00 with your user exit. The IMS initialization exit (DFSINTX0) cannot access MSC control blocks during IMS initialization, because the MSC control blocks are not built until restart. If the DFSINTX0 exit tries to access MSC control blocks, it will not find any. The MSC control blocks that cannot be found are LLB, LCB, LNB, and RCNT. The DFSMSCE0 user exit initialization entry point (which is called at IMS restart) and the other entry points can access those control blocks with FIND/SCAN control block callable services. See the prolog of that user exit for details and samples of those services.
Sample routine location	IMS.ADFSSMPL (member name DFSINTX0).

About user data areas

The user data areas can be used to provide access to user tables that can then be referenced by any user exit that has access to the data area. An example of the use of general user data area is for ETO. You can use the general user data area to define access limits for terminals or users by total number, department, time of day, or other criteria. You can also use the data area to define LTERM-to-user or user-to-terminal relationships to aid your installation logon and signon exit routine processes.

For APPC, you can use the LU6.2 user data area along with the LU6.2 User Edit exit routine to emulate MFS. To do so, the LU6.2 user data area is built by DFSINTX0 to hold a list of LTERM and MOD names available to the I/O PCB. IMS then passes the address of the LU6.2 user data area LU 6.2 Edit exit routine for input and output messages from a LU6.2 destination. The LU 6.2 Edit exit routine can use the list of LTERM names to redirect output to a non-LU6.2 destination, or the list of MOD names to format a message.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

The following table shows the IMS initialization exit parameters. The address of this parameter list is in the IMS standard user exit parameter list field SXPLFSPL. The Initialization exit routine parameter list is mapped by macro DFSINTXP.

Table 65. IMS initialization exit parameter list

Offset	Length	Description
+0	4	CLB address
+4	4	SCD address
+8	4	0, as an indication that no user table exists
+12	4	0, as an indication that no LU 6.2 user table exists

Table 65. IMS initialization exit parameter list (continued)

Offset	Length	Description
+16	1	Input/Output Flag Byte **X'80'** **0** No password verification (default). To enable password verification, set this flag to 1. **X'40'** **0** Default ETO ALOT=0 process **X'10'** **0** Static ISC resource sharing (default) **X'08'** **0** ETO LU type 3 is not allowed to log on as a SLU1 (default) **X'04'** **0** ETO LU type 3 is not allowed to log on as a 3270 printer (default)

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains the return code.

The address of the general user data area created by this exit routine can be returned in the Initialization exit parameter list at +8. If zero, no general user data area was created. If non-zero, IMS saves the address in the SCD control block at SCDINTXP.

The address of the LU 6.2 user table created by this exit routine can be returned in the Initialization exit parameter list at +12. If zero, no LU 6.2 user table was created.

Register	Contents	
15	One of the following return codes:	
	Return code	**Meaning**
	0	Initialization of IMS continues.
	4	Regardless of ETO specification, ETO terminal support is not required. Message DFS3648 is sent to the system console. Setting RC=4 resets both ETO function and logon user data support.
	8	Regardless of ETO specification, ETO terminal support is not required but logon user data is supported for static terminals. Message DFS3648 is sent to the system console. Setting RC=8 resets ETO function only.

Table 66. IMS initialization exit parameter list

Offset	Length	Description
+16	1	Input/Output Flag Byte **X'80'** **0** No password verification (default) **1** Enable password verification **X'40'** **0** Default ETO ALOT=0 process **1** Alternate ETO ALOT=0 process **X'20'** **0** Default VGR for ISC **1** Disable VGR for ISC **X'10'** **0** Normal static ISC resource sharing (default) **1** Disable resource sharing for static ISC terminals in the IMSplex **X'08'** **0** ETO LU type 3 is not allowed to log on as a SLU1 (default) **1** ETO LU type 3 is allowed to log on as a SLU1[“1” on page 187] **X'04'** **0** ETO LU type 3 is not allowed to log on as a 3270 printer (default) **1** ETO LU type 3 is allowed to log on as a 3270 printer[“1” on page 187]

Notes:

1. ETO LU type 3 is allowed to log on either as SLU1 or 3270 printer, but not both.

Related tasks

Using the MOD name and LTERM interface (Communications and Connections)

Related reference

“LU 6.2 Edit exit routine (DFSLUEE0)” on page 202

The LU 6.2 Edit exit routine (DFSLUEE0) enables you to edit input and output LU 6.2 messages for IMS-managed LU 6.2 conversations. It is also called if a message is inserted from an alternate PCB destined for an LU 6.2 destination.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Input Message Field edit routine (DFSME000)

Use the Input Message Field edit routine (DFSME000) to perform common editing functions and simplify programming.

This topic contains Product-sensitive Programming Interface information. \

This topic describes how to write an Input Message Field edit routine. Because this routine is usually used with the Input Message Segment edit routine, you'll find references to both routines throughout the following paragraphs.

Subsections:

About this routine

MFS application designers should consider the use of Input Message Field and Segment edit routines to perform common editing functions such as numeric validation or conversion of blanks to numeric zeros. Field and Segment edit routines can simplify programming by using standard field edits to perform functions that would otherwise have to be coded in each application program.

The following table shows the attributes of the Input Message Field Edit routine.

Table 67. Input message field edit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSME000.
Binding	A Field edit routine must have a CSECT name of DFSMEnnn, where nnn is a number from 001 to 126 that corresponds with the routine number specified in the MFLD statement. In order for IMS to properly load the edit routine during IMS initialization, move the edit routine to an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. The Field edit routine can only modify the data in the field created by MFS and must not cause any waits.
Including the routine	Move the edit routine to an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL, or ensure that the library on the IMSGEN macro is added to the JOBLIB, STEPLIB, or LINKLIST concatenation.

Table 67. Input message field edit routine attributes (continued)

Attribute	Description
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSI00) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for IMS callable services. Manually link this exit with DFSCSI00 to use callable services. No additional linking is required to use IMS callable services.
Sample routine location	IMS.SDFSSMPL (member name DFSME000).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the edit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
9	Address of CLB/ECB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

Description of parameter list format

IMS.ADFSMAC contains a DSECT of the parameter list addressed by register 1 (use COPY MFSFLDE) as follows:

Byte	Contents	
0		
	Bit	**Contents**
	0,1	Message formatting option: • 00 = option 1 • 01 = option 2 • 11 = option 3
	2	Zero (Field edit routine)
	3	Reserved
	4	1 if the first 2 bytes in the field contains attribute information
	5	1 if the field contains extended field attribute information
	6	Reserved
	7	Reserved
1	Zeros	

Byte	Contents
2	The number of reserved extended field attribute bytes in the field. These bytes appear immediately after the 3270 attribute bytes, if any.
3	The entry vector in binary (0 to 255).
4-7	The execute length (length-1) of the field as defined in the MFLD statement. If ATTR=YES is specified, this field contains (length-3).
8-11	The field address after MFS editing (before uppercase translation and null compression for option 1 and 2 fields). If ATTR=YES is specified, this is the address of the first data byte after the two attribute bytes. For option 3, this is the address of the 2-byte field length, which begins the completed option 3 field.

Contents of registers on exit

Before returning to IMS, the edit routine must restore all registers except for register 15, which must contain one of the following return codes:

Register	Contents
15	Return code value from 0 to 255

Function of the sample routine

The functions of this IMS-supplied routine are as follows:

Vector	Resulting action
0	Converts blanks to zoned decimal zeros (X'F0').
1	Converts blanks to zoned decimal zeros (X'F0') and replaces non-zoned decimal characters with a question mark (?). If ? is inserted, the routine sets a return code of 8 and, if an attribute (ATTR) area is present, sets the CURSOR,HI attributes.
2	Converts the binary cursor address field to zoned decimal if its length is 4 bytes. If the field is not 4 bytes, a return code of 8 is set.
>2	Sets a return code equal to the entry vector (if the vector is greater than 2).

This routine will handle option 1, 2, and 3 formats. For option 1 and 2, MFLD FILL=NULL and an entry vector of 1 can produce undesirable results.

Related reference

“Input Message Segment edit routine (DFSME127)” on page 191
The Input Message Segment edit routine (DFSME127) can be used by MFS application designers to perform common editing functions such as numeric validation or conversion of blanks to numeric zeros. Field and Segment edit routines can simplify programming by using standard field edits to perform functions that would otherwise need to be coded in each application program.

“IMS callable services” on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Calling the Input Message Field edit routine

Call the Input Message Field edit routine after MFS editing.

Field edit routines are given control after MFS editing (before Segment edit routines, uppercase translation for all options, and null compression for option 1 or 2). The routine can validate or alter

the data and pass a return code to MFS. MFS maintains the highest return code of all Field edit routines for each segment and passes that code to the Segment edit routine after all fields for that segment are edited.

Defining edit routines

Assign routine numbers and entry vectors for the Input Message Field edit routine in the MFSEXITF parameter in the IMS startup parameters.

Field edit routines are defined in the MID's MFLD statements in terms of a routine number and entry vector.

Routine numbers identify the routine to be used for this field/segment. Routine numbers range from 000 to 127. IMS-provided routines use numbers 000 (field edit, DFSME000) and 127 (segment edit, DFSME127).

If you are using both the Field edit and Segment edit routines with your IMS system, the Field edit routine should be assigned routine numbers that are lower than the numbers assigned for the Segment edit routine. Therefore, the Field edit number should be a decimal number greater than or equal to 0, and less than the default or specified value for the Segment edit routine number parameter. The default for the Field edit routine is 0.

An installation standard should be established regarding the assignment of routine numbers. For example, you could assign Field edit routines numbers in ascending sequence from 001 to 063 (and if you're using Segment edit routines as well, assign them numbers in descending sequence from 126 to 064).

Recommendation: Assign lower numbers to field exit routines and higher number to segment exit routines.

Entry vectors are passed to the edit routine when it is activated. Entry vector values can range from 0 to 255. The entry vector value can be thought of as an additional qualification of the routine to be activated. For example, routine number 025 can perform numeric validation of a field; entry vector 0 can replace leading blanks with zeros, and entry vector 1 can perform numeric validation.

If data is entered from the terminal in lowercase, the data is in lowercase when it is presented to the edit routine. If data in an input segment is in nongraphic form, GRAPHIC=NO should be specified in the SEG statement to prevent null compression and uppercase translation. A valid byte value of a binary field could be equivalent to a null character (X'3F') or some lowercase alphanumeric (for example, a=X'81'). In this case, GRAPHIC=NO should be specified.

Related Reading: For a description of which characters MFS considers graphic, see the SEG statement section in *IMS Version 15.2 System Utilities*.

Related information

COMM macro statement (System Definition)

Performance considerations

When Field and Segment edit routines are used, extra processing occurs in the IMS control region and, if used extensively, a measurable performance cost is incurred.

These edit routines also can improve performance by reducing processing time in the message processing region, by reducing logging and queuing time, and by allowing field verification and correction to be accomplished without scheduling an application program. Efficiency of these user-written routines should be a prime concern. Because these routines execute in the IMS control region, an abend in the edit routine causes the IMS control region to abend.

Input Message Segment edit routine (DFSME127)

The Input Message Segment edit routine (DFSME127) can be used by MFS application designers to perform common editing functions such as numeric validation or conversion of blanks to numeric zeros.

Field and Segment edit routines can simplify programming by using standard field edits to perform functions that would otherwise need to be coded in each application program.

This topic contains Product-sensitive Programming Interface information.

This topic describes how to write an Input Message Segment edit routine. Because this routine is usually used with the Input Message Field edit routine, you will find references to both routines throughout the following paragraphs.

Subsections:

About this routine

The following table shows the attributes of the Input Message Segment edit routine.

Table 68. Input message segment edit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSME127.
binding	A Segment edit routine must have a CSECT name of DFSMEnnn, where nnn is a number from 001 to 126 that corresponds with the routine number specified in the SEG statement. It must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
Including the routine	This exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for IMS callable services. Manually link this exit with DFSCSI00 to use callable services. No additional linking is required to use IMS callable services.
Sample routine location	IMS.ADFSSRC (member name DFSME127)

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the edit routine.

Contents of registers on entry

On entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Address of CLB.
1	Address of parameter list.
9	Address of CLB/ECB.
13	Address of save area. The edit routine must not change the first three words.

Register	Contents
14	Return address to IMS.
15	Entry point of edit routine.

Description of parameter list format

IMS.ADFSMAC contains a DSECT of the parameter list addressed by register 1 (use COPY MFSSEGE) as follows:

Byte	Contents		
0			
	Bit	**Contents**	
	0, 1	Message formatting option: 00 = option 1 01 = option 2 11 = option 3	
	2	1 (Segment edit routine)	
	3		
		1	If this message can be routed back to the device by specifying return code 16. This bit is set on when the following conditions are met: • PAGDEL=YES or OPTIONS=(...,PAGDEL,...) is specified in the TERMINAL macro for this device. • The device has an output logical terminal. If the message contains a valid operator logical paging request, bit 3 can be set on. However, this message is not returned to the terminal if requested.
	4-7	Reserved	
1,2	Zeros		
3	The entry vector is binary (0 to 255).		
4-7	The maximum segment length.		
8-11	The segment address.		
12-15	The highest return code from the Field edits for this segment.		
16-23	The next MOD name.		

The Segment Edit routine can modify only the segment contents, the save area, and the next MOD name field of the parameter list. The MOD name field name should be changed when the edit routine returns the input message to the device. If the segment is option 1 or 2, the routine can set the segment length field to any value from 0 to the maximum segment length. The Segment Edit routine must not cause any waits.

Contents of registers on exit

On return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes:

Return code	Meaning
0	Continue processing.

Return code	Meaning
4	Cancel this segment.
8	Cancel this message (IMS sends the message DFS298 `INPUT MESSAGE CANCELED BY MFS EXIT`).
12	Cancel this message and return to the user the message whose number is in register 1.
16	Return this message to the input device. This code is allowed only when bit 3 of byte 0 in the parameter list is set on.

All segments of a multisegment message are edited before the message is returned to the device (return code 16); if return code 8 or 12 is specified for a segment other than the final one, the message is canceled immediately and the remaining segments are not edited.

In IMS releases with ETO, the Input Message Segment edit routine cannot use return code 16 during the ETO signon process. This is due to the lack of a valid output LTERM.

Function of the sample routine

The functions of this routine are based on the entry vector and the highest Field edit routine return code (FLD-RC) for the segment. This routine only performs modifications of messages using formatting options 1 and 2. The functions are shown in the following table.

Table 69. Input message segment edit routine functions based on the entry vector

Input vector	FLD-RC	Resulting function action	SEG-RC
0	< 4	None.	0
	>= 4	Places EBCDIC return code in last 3 bytes of the segment.	0
1	< 4	None.	0
	>= 4	Places EBCDIC return code in last 3 bytes of the segment.	0
	< 8	None.	4
2	< 4	None.	0
	=4 <8	Places EBCDIC return code in last 3 bytes of the segment.	0
	>= 8	None.	8
3	<4	None.	0
	=4 <8	Places EBCDIC return code in last 3 bytes of the segment.	
	>=8	None.	6
4	ANY	Sets FLD-RC as user message number.	12

Notes:

1. To continue processing
2. To cancel this segment
3. To cancel this message
4. To send this message back to the entering terminal
5. To cancel this message and send the user message, whose number is in register 1, back to the entering terminal

Related reference

"Input Message Field edit routine (DFSME000)" on page 188
Use the Input Message Field edit routine (DFSME000) to perform common editing functions and simplify programming.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

Calling the Input Message Segment edit routine

Segment edit routines are given control when all the MFS editing and editing by Field edit routines is complete for a message (before uppercase translation, but after null compression for messages using option 1 and 2, and after field sort for option 3 messages).

Based on the return code received from Field or Segment edit routine, the Segment edit routine can:

- Continue processing.
- Modify the segment.
- Cancel the segment.
- Cancel the message and IMS will notify the operator using the message `DFS298 INPUT MESSAGE CANCELED BY MFS EXIT`.
- Return a predefined message to the terminal.
- Return the input message to the terminal.

Restriction: The following applies only to IMS releases with ETO. During the ETO dynamic terminal signon process, the Input Message Segment edit routine cannot use return code 16 to return the input message to the terminal. This is because a valid output LTERM has not yet been established.

Defining edit routines

Assign a routine number and an entry vector for the Input Message Segment edit routine in the MFSEXITS parameter in the IMS startup parameters.

Segment edit routines are defined in the MID's SEG statements. Each routine is defined in terms of a routine number and an entry vector.

Routine numbers identify the routine to be used for this field or segment. Routine numbers range from 000 to 127. IMS-provided routines use numbers 000 (Field edit, DFSME000) and 127 (Segment edit, DFSME127).

If you are using both the Field edit and Segment edit routines with your IMS system, the Field edit routine should be assigned routine numbers lower than the numbers assigned for the Segment edit routine. Therefore, the Field exit number should be a decimal number greater than or equal to 0, and less than the default or specified value for the Segment exit routine number parameter. The default for the Field edit routine is 0.

An installation standard should be established regarding the assignment of routine numbers. For example, you could assign Segment edit routines numbers in descending sequence from 126 to 064 (and if you're using Field edit routines as well, assign them numbers in ascending sequence from 001 to 063).

Recommendation: Assign lower numbers to Field edit routines and higher numbers to Segment edit routines.

Entry vectors are passed to the edit routine when it is activated. Entry vector values can range from 0 to 255. The entry vector value can be thought of as an additional qualification of the routine to be activated.

For example, routine number 025 can perform numeric validation of a field; entry vector 0 can replace leading blanks with zeros, and entry vector 1 can perform numeric validation.

If data is entered from the terminal in lowercase, the data is in lowercase when it is presented to the edit routine. If data in an input segment is in nongraphic form, GRAPHIC=NO should be specified in the SEG statement to prevent null compression and uppercase translation. A valid byte value of a binary field could be a null character (X'3F') or some lowercase alphanumeric (for example, a=X'81'). In this case, GRAPHIC=NO should be specified.

Related reference

SEG statement (System Utilities)

Related information

COMM macro statement (System Definition)

Performance considerations

Efficiency of the Input Message Segment edit routine should be a prime concern.

When Field and Segment edit routines are used, extra processing occurs in the IMS control region and, if used extensively, a measurable performance cost is incurred. At the same time, these edit routines can improve performance by reducing processing time in the message processing region, by reducing logging and queuing time, and by allowing field verification and correction to be accomplished without scheduling an application program.

Logoff exit routine (DFSLGFX0)

The Logoff exit routine handles all non-MSC, non-LU 6.2 VTAM nodes with which IMS communicates.

This topic contains Product-sensitive Programming Interface information.

This topic describes how you can use the Logoff exit routine to perform processing that complements the Logon exit routine (DFSLGNX0).

Subsections:

About this routine

IMS calls the Logoff exit routine for all non-MSC, non-LU 6.2 VTAM nodes with which IMS communicates and for all master terminal operator (MTO) logoffs, even if it did not call the Logon exit routine for the MTO at logon. (Keep this in mind if your installation maintains a logon count.) All attempts to log off of ACF/VTAM terminals cause IMS to call this exit routine.

Recommendation: Although the Logon exit routine and the Logoff exit routine are optional, if you include one, you should also include the other to perform any necessary cleanup operations.

The following table shows the attributes of the Logoff exit routine.

Table 70. Logoff exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSLGFX0.
Including the routine	If you want IMS to call this exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. If the Logoff exit routine is included, IMS automatically loads it each time IMS is initialized.

Table 70. Logoff exit routine attributes (continued)

Attribute	Description
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSI00) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found at offset 0 of the Logoff user exit parameter list. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.ADFSSMPL (member name DFSLGFX0).

Extended Recovery Facility (XRF) considerations

Each time IMS calls the Logoff exit routine, the exit routine receives information on the XRF status of IMS. IMS calls the exit routine if XRF tracking fails.

Resetting the significant status

You can use this exit to reset the significant status for a terminal in one of the following states:

Conversational
Exclusive
Test
Preset
MFS test
Full-function response
Fast Path response

Note: Test and preset states are nonrecoverable, so IMS resets the significant status automatically.

A parameter passed to the exit routine indicates the status of the terminal or ETO user at signoff. All users except ETO terminals can reset the status in the output parameters.

For conversation mode, IMS performs the equivalent of an `/EXIT` command for the conversation.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

The following table lists the logoff exit parameters. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 71. Logoff exit parameter list

Offset	Length	Description
+0	4	Current ECB address
+4	4	SCD address
+8	4	Address of User Table
+12	4	Address of the STATUS_IN and STATUS_OUT vectors. The status vectors are mapped by the DFSSTCHK macro. For the contents of the STATUS_IN vector see the following table.

Contents of STATUS_IN

The input status vector is a two-byte field that indicates the significant status of a terminal when the exit routine is called. The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status as follows:

Value	Description
X'80'	Conversation
X'40'	Exclusive
X'20'	Test
X'10'	Preset
X'08'	MFS test
X'04'	Full-function response
X'02'	Fast Path response

Contents of STATUS_OUT

The output status vector is a two-byte field that indicates changes to the terminal's significant status made by the exit routine. IMS uses the contents of STATUS_OUT as an indicator to exit a conversation and reset significant status. The default for this field is zeros, indicating that no significant status is reset.

The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status as follows:

Value	Description
X'80'	Exit conversation
X'40'	Reset exclusive
X'20'	Reset test
X'10'	Reset preset
X'08'	Reset MFS test
X'04'	Reset full-function response
X'02'	Reset Fast Path response

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15. The content of registers on exit is as follows:

Register	Contents
15	Ignored by IMS in all cases.

Related reference

"Logon exit routine (DFSLGNX0)" on page 199
The Logon exit routine (DFSLGNX0) handles all non-MSC, non-LU 6.2 VTAM nodes (excluding MTOs at IMS initialization) with which IMS communicates. The Logon exit routine enables you to control the way logons are processed.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Logon exit routine (DFSLGNX0)

The Logon exit routine (DFSLGNX0) handles all non-MSC, non-LU 6.2 VTAM nodes (excluding MTOs at IMS initialization) with which IMS communicates. The Logon exit routine enables you to control the way logons are processed.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

The exit routine must handle all non-MSC, non-LU 6.2 VTAM nodes (excluding MTOs at IMS initialization) with which IMS communicates. All attempts to log on to ACF/VTAM terminals if ETO is active cause IMS to call this exit routine.

Depending on your installation's needs, you can write the Logon exit routine to:

- Select the logon descriptor that you want IMS to reference when building the terminal control block structure for the logical unit (LU) that is logging on.
- Create or modify the user data that you want IMS to pass to the Signon exit routine (DFSSGNX0). The user data can be entered as autologon data, with the /OPNDST command, or with the VTAM internal commands INITSELF or INITOTHER. Alternatively, the Logon exit routine can build the user data.
- Allow or disallow a logon attempt based on the maximum number of sessions, or manage logons according to the time of day, certain terminal names, or other criteria that you specify.
- Specify or override the autologoff (ALOT), autosignoff (ASOT), screen size, or model values.
- Override the AUTOSIGN and NOAUTSGN keywords for static terminals.
- Override the default status recovery mode for the following terminals:
  - Static terminals
  - SLUP dynamic terminals
  - FINANCE dynamic terminals
  - ISC dynamic terminals

The Logon exit routine is optional.

Recommendation: If you include this exit routine, you should also include the Logoff exit routine (DFSLGFX0) to perform any necessary cleanup operations.

If you do not supply the Logon exit routine, logons proceed as usual with the chosen logon descriptor.

The following table shows the attributes of the Logon exit routine.

Table 72. Logon exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSLGNX0.
Including the routine	If you want IMS to call the Logon exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. If the exit routine is included, IMS automatically loads it each time IMS is initialized if ETO=Y (after the Initialization exit routine, DFSINTX0, changed the ETO= keyword).
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the current address ECB found at offset 0 for the DFSCSII0 call. • Link DFSCSI00 with your user exit. **Restriction:** Global terminal or user resource information is not available to user exit DFSLGNX0. Callable services will only return local information for DFSLGNX0.
Sample routine location	IMS.ADFSSMPL

Extended Recovery Facility (XRF) considerations

During XRF tracking mode, IMS calls the Logon exit routine in the alternate system when the terminal control blocks are created for an XRF type 1 session with an ETO terminal. If processing is on an XRF alternate system, IMS ignores the contents of register 15 on exit. The exit routine is called during XRF alternate tracking only for the logon of a class 1 terminal.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

The following table lists the user logon parameters. The mapping for this parameter list is DSECT LGNXPARM in DFSLGNXP macro. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 73. User logon exit parameter list

Offset	Length	Description
+0	4	Current ECB address.
+4	4	SCD address.
+8	4	Pointer to User Table.
+12	4	Pointer to the parameter list received from ACF/VTAM when application logon or SCIP bind exit routines are scheduled. If processing is on an XRF system, this value is zero.
+16	4	Pointer to multi-word parameter list, mapped by DSECT LGNXPARM in DFSLGNXP macro.
+20	4	CLB pointer for the node trying to logon. If the node does not yet exist, this value is zero. The node always exists on an XRF system.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains one of the following return codes:

Register	Contents	
15	One of the following return codes:	
	Return code	**Meaning**
	0	LOGON accepted
	4	LOGON rejected

Related reference

"Logoff exit routine (DFSLGFX0)" on page 196
The Logoff exit routine handles all non-MSC, non-LU 6.2 VTAM nodes with which IMS communicates.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

z/OS: LOGON exit

Selecting a logon descriptor

If the terminal control block structure already exists for the terminal that is logging on, no logon descriptor is needed, and IMS uses the existing terminal control block structure.

If no terminal control block structure exists for the terminal, you can write the Logon exit routine to select the logon descriptor, select a logon descriptor by using the LOGOND= keyword, or let IMS select the logon descriptor using the LU name or default descriptor.

The following figure shows the search order IMS uses to select the logon descriptor. IMS selects the first valid logon descriptor that it finds and uses that logon descriptor to build the terminal control block

structure. If IMS cannot find a valid logon descriptor, including the default logon descriptor, it rejects the logon request.

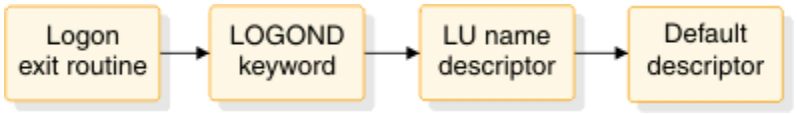


Figure 13. Logon descriptor search order

If the exit routine supplies the name of a valid logon descriptor, IMS uses the logon descriptor associated with that name to build the terminal control block structure. If the Logon exit routine does not choose a logon descriptor, or if the exit routine is not included in the system, IMS uses the logon descriptor requested on the LOGOND= keyword (entering the keyword and descriptor as user data when you log on). If neither the exit routine nor the LOGOND= keyword identifies a valid logon descriptor, IMS searches for a logon descriptor with the same name as the logical unit (LU). If IMS cannot locate a logon descriptor with this name, IMS uses the default logon descriptor table shown in the following table to select the logon descriptor.

Table 74. Default logon descriptor table

CINIT LUTYPE	CINIT TS	Default logon descriptor
X'06'	Not applicable	DFSLU61
X'04'	Not applicable	DFSSLU4
X'02'	Not applicable	DFSSLU2
X'01'	Not applicable	DFSSLU1
X'00'	X'04'	DFSSLUP
X'00'	X'03'	DFS3270

IMS cannot generate DFSFIN or DFSNTO logon descriptors because of conflicting CINIT information. The wrong default logon descriptors are chosen for the FINANCE and NTO terminal types unless you do all of the following:

- Write the Logon exit routine so that it always supplies the appropriate logon descriptor name.
- Rename DFSFIN to DFSSLUP if no SLU P terminals exist.
- Rename DFSNTO to DFSSLU1 if no SLU1 terminals exist.

If you do not want dynamic logons for a certain LU type, delete the default logon descriptor for that type from the system, and be sure that the exit routine does not attempt to choose it.

Regardless of how the logon descriptor is selected, the descriptor must agree with the LUTYPE and TS fields (in the MODEENT macro of the VTAM mode table), or IMS rejects the logon request.

LU 6.2 Edit exit routine (DFSLUEE0)

The LU 6.2 Edit exit routine (DFSLUEE0) enables you to edit input and output LU 6.2 messages for IMS-managed LU 6.2 conversations. It is also called if a message is inserted from an alternate PCB destined for an LU 6.2 destination.

This topic describes the LU 6.2 Edit exit routine. This exit routine is for use with standard IMS and modified IMS application programs. It is not called for CPI Communications driven application programs.

Subsections:

- "About this routine" on page 203
- "Communicating with IMS" on page 204

About this routine

You can write the LU 6.2 Edit exit routine to:

- Change the APPC local LU name of an asynchronous LU 6.2 outbound conversation.
- Change the synchronization level of an asynchronous LU 6.2 conversation.
- View the contents of a message segment and continue processing.
- Change the contents of a message segment and continue processing.
- Discard a message segment.
- Perform a DEALLOCATE_ABEND of the LU 6.2 conversation.

For input messages, IMS calls the LU 6.2 Edit exit routine for each message segment before the message segment is inserted to the IMS message queue. The exit routine can edit message segments as necessary before the application program processes the input message.

For output messages, IMS calls the LU 6.2 Edit exit routine for each message segment before the message segment is sent to the LU 6.2 program. The exit routine can intercept the data sent by the application program and edit it for the particular destination.

The following table shows the attributes of the LU 6.2 Edit exit routine.

Table 75. LU 6.2 edit exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSLUEE0.
Binding	The LU 6.2 Edit exit routine must be reentrant. The IMS-provided default exit routine specifies a return code of zero. If you write your own exit routine, replace the IMS default routine by binding the one you wrote into the IMS.SDFSRESL or including it in an authorized library in the JOBLIB, STEPLIB, or LINKLIB library concatenated in front of IMS.SDFSRESL.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	IMS.ADFSSRC (member name DFSLUEE0). This sample is a default exit routine, which IMS always calls for LU 6.2 messages processed under the DL/I call interface.

Changing a message segment

The LU 6.2 Edit exit routine can change the message length and contents, provided that it resets the message length field to reflect the new length. The exit routine can increase the message length by up to 256 bytes, but the total length (length field, flag field, and message) cannot exceed 32,767 bytes. If the message exceeds this limit, IMS truncates the message and issues DFS1967 to the master terminal operator (MTO) to indicate a message buffer overlay. The exit routine can reduce the message length without restriction.

Changing a local LU name

The LU 6.2 Edit exit routine can change the local LU name. Word 12 points to the local LU name that is used to allocate outbound conversations. The LU 6.2 Edit exit routine can be used to change that name. The local LU name can be changed only for outbound conversations.

Network-qualified names

Network-qualified LU names can be up to 17 bytes long.

MOD name support for APPC

An LU 6.2 application program can send the LTERM and the MOD name in the first segment of the message. IMS saves the LTERM and MOD name in the I/O PCB.

At entry, IMS provides the address of the MOD name in the first segment of the message sent to the LU 6.2 Edit exit routine (DFSLUEE0). DFSLUEE0 checks the contents of the first message segment. If IMS finds the MOD name, it uses the MOD name to format the output message. If IMS finds the LTERM, it can use the LTERM to change the destination of the output.

Use the Initialization exit routine (DFSINTX0) 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.

Communicating with IMS

IMS uses the entry and exit registers and a parameter list to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

<table>
<tr><th>Register</th><th>Contents</th></tr>
<tr><td>1</td><td>Address of parameter list. The parameter list contains the following addresses.
<table>
<tr><th>Bytes</th><th>Content</th></tr>
<tr><td>00-03</td><td>Address of a flag field indicating what type of message caused IMS to call the exit routine. This field contains one of the following flags (fixed length, right justified, padded with zeros):
<table>
<tr><td>0</td><td>Input message</td></tr>
<tr><td>4</td><td>Output message</td></tr>
</table></td></tr>
<tr><td>04-07</td><td>Address of the area containing either the input or output message segment length, message flag, and message segment (variable length, left justified). The value in the length field includes the length field, flag field, and message.</td></tr>
<tr><td>08-11</td><td>Address of transaction code (fixed length, left justified, padded with blanks).</td></tr>
<tr><td>12-15</td><td>Address of LU name (fixed length, left justified, padded with blanks).</td></tr>
<tr><td>16-19</td><td>Address of user ID (fixed length, left justified, padded with blanks).</td></tr>
<tr><td>20-23</td><td>Address of return code, which is an exit parameter.</td></tr>
<tr><td>24-27</td><td>Address of LTERM (fixed length, left justified, padded with blanks).</td></tr>
<tr><td>28-31</td><td>Address of MOD name (fixed length, left justified, padded with blanks).</td></tr>
<tr><td>32-35</td><td>Address of user table, which is an entry parameter.</td></tr>
<tr><td>36-39</td><td>Address of message flag (if bit zero of the message flag equals 1, it is the first segment).</td></tr>
<tr><td>40-43</td><td>Address of user ID indicator byte, which describes the content of the user ID field and can have a value of one of the following: U (user ID), L (LTERM), P (PSBname), or O (Other).</td></tr>
</table></td></tr>
</table>

Register	Contents	
	44–47	For asynchronous outbound conversations the exit can change the address of the synchronization level (one byte). The synchronization level can be N (None), C (Confirm), or S (Syncpoint). For asynchronous conversations the exit can change the synchronization level. Note that only synchronization level N and C are supported for asynchronous conversations.
	48-52	Address of the local LU name (8 bytes) or the base LU if no local LU name has been used. For asynchronous outbound conversations, the exit can change it to another LU defined for this IMS.
13	Address of save area. The exit routine must not change the first three words.	
14	Return address to IMS.	
15	Entry point of exit routine.	

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers. The registers contain the following:

Register	Contents	
1	Address of parameter list (provided at entry). The parameter list contains the following addresses.	
	Bytes	**Content**
	00-03	Used on entry only.
	04-07	Address of the area containing the message segment length, message flag, and message segment (variable length, left justified). The value in the length field is the total length and includes the length field, flag field, and message.
	08-19	Used on entry only.
	20-23	Address of the area for one of the following return codes from the exit routine. (IMS treats any other value as 0.) **0** IMS performs the default action: continue processing. **2** For asynchronous conversations, IMS must discard the message if it is not deliverable. **4** Discard this message segment. **8** DEALLOCATE_ABEND the conversation.
	24-27	Address of LTERM (exit parameter).
	28-31	Address of MOD name (entry and exit parameter).
	32-35	Address of User Table (entry parameter).
	36-39	Address of message flag (Bit 0 = 1 then first segment) (entry parameter).
	40-43	Address of user ID indicator.

Register	Contents	
	44-47	For asynchronous outbound conversations the exit can change the address of the synchronization level (one byte). The synchronization level can be N (None), C (Confirm), or S (Syncpoint). For asynchronous conversations the exit can change the synchronization level. Note that only synchronization level N and C are supported for asynchronous conversations.
	48-52	Address of the local LU name (8 bytes) or the base LU if no local LU name has been used. For asynchronous outbound conversations, the exit can change it to another LU defined for this IMS.

Data format of parameters

The following table shows the data type, length, and format of the fields to which the parameter list (addressed by register 1) points.

Table 76. Format of parameters

Bytes	Data address	Parameter use	Data type	Data length	Data format[1]
00-03	Address of flag	Fixed length, right justified, padded with zeros	Input	4 bytes	X'*flag*'
04-07	Address of message segment length, message flag, and message segment	Variable length, left justified	Input and output	n bytes[2]	LLZZ*message*
08-11	Address of transaction code	Fixed length, left justified, padded with blanks	Input	8 bytes	*code*bbbb
12-15	Address of LU name	Fixed length, left justified, padded with blanks	Input	17 bytes	*name*bbbb
16-19	Address of user ID	Fixed length, left justified, padded with blanks	Input	8 bytes	*user ID*bb
20-23	Address of return code	Fixed length, right justified, padded with zeros	Output	4 bytes	X'*code*'
24-27	Address of LTERM	Fixed length, right justified, padded with zeros	Output	8 bytes	*ltermname*
28-31	Address of MOD name	Fixed length, left-justified, padded with blanks	Input and output	8 bytes	*modname*
32-35	Address of user table	Variable length	Output	? bytes[3]	*usertablename*
36-39	Address of message flag	Fixed length	Output	1 byte	X'*code*'
40-43	Address of user ID indicator	Fixed length	Input	1 byte	*indicator*

Table 76. Format of parameters (continued)

Bytes	Data address	Parameter use	Data type	Data length	Data format[1]
44–47	Address of synchronization level	Fixed length	Input and output	1 byte	APPC synchronization level
48-52	Address of the local LU name	Fixed length	Input and output	8 bytes	APPC local LU name

Note:

[1]**ZZ** = flag field; **LL** = length field; bb = blanks; words in *italics* represent data values. The value in the length field **LL** includes the length field, flag field, and message.

[2]The exit routine can increase the message length by up to 256 bytes, but the total length cannot exceed 32,767 bytes.

[3]The length of this user table is determined by the user.

Related tasks

Qualifying network LU names (Communications and Connections)

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Message Control/Error exit routine (DFSCMUX0)

You can use the Message Control/Error exit routine (DFSCMUX0) to control transactions, responses, and message switches that are in error.

This topic contains Product-sensitive Programming Interface information.

This topic describes the Message Control/Error exit routine. The exit routine can request that IMS handle the messages that are in error, depending on the condition that led IMS to call the exit routine. The /DEQUEUE command supports the MSNAME keyword so that this control is extended to messages queued on Multiple Systems Coupling (MSC) links.

Subsections:

About this routine

You can write the Message Control/Error exit routine to:

- Perform processing at MSC link start and link termination time that is unique to your installation, such as obtaining and freeing additional storage, and activating and deactivating a program.
- Reroute a message to a different local or remote transaction, local or remote LTERM, or an LU 6.2 destination. The target LTERM must be an existing LTERM; IMS does *not* dynamically create the LTERM, even if the Extended Terminal Option (ETO) feature is active. For more information about the ETO feature, see Overview of the Extended Terminal Option (Communications and Connections).
- Discard a message and send an informational message to the current master terminal operator (MTO) or input terminal to indicate that the message is discarded.
- Suppress the /DEQUEUE command, or suppress the command and send an informational message to the entering to indicate that the command is suppressed.

- Process late or redundant response messages that are sent in response to a synchronous program switch request. A late response message is any message that is sent after the original request timed out. A redundant response message is any message that is sent after the request receives the first response. The default action for a late or redundant response message is to dequeue it. You can write a DFSCMUX0 exit routine to route late or redundant response messages to a logical terminal or an OTMA destination instead.

A sample exit routine is available from the IMS library. The sample exit routine is the default routine. IMS calls the sample exit routine unless you replace it with your own version. The sample exit routine includes code that supports the following keywords on the /DEQUEUE command:

lterm
node
msname
luname plus *tpname*

The default action for this exit routine is to proceed with the /DEQUEUE command.

The following table shows the attributes of the Message Control/Error exit routine.

Table 77. Message control/error exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCMUX0.
Binding	This exit routine must be reentrant. The sample exit routine is a default routine. If you write your own exit routine, you must bind it with the IMS control region SDFSRESL.
IMS callable services	This exit routine cannot use callable services.
Sample routine location	IMS.SDFSSMPL (member name DFSCMUX0). The sample routine provided is compatible with the MSC error handling and /DEQUEUE command processing that exists for prior releases of IMS. You can ensure compatibility by including this sample exit routine logic in your customized version. The sample exit routine prolog contains additional usage information. The MSNB DSECT is located in IMS.SDFSSMPL (member name MSNB).

Calling the routine

IMS calls the Message Control/Error exit routine and sets an entry flag in the interface block as a result of one of the following:

- Link start.

  A RSTART LINK command is entered to start an MSC link or when the MSC link is started by the partner system (MSC environment only).
- Link termination.

  This exit routine is called at link termination time mainly when a PSTOP link command is entered from IMS, or the link is stopped by the partner IMS, for all access methods of MSC. Most errors (such as, invalid data, queue error, or access method) in MSC do not cause the link to be terminated.

  For MSC VTAM, the exit routine is also called in the following cases:

  – CLSDST/TERMSESS complete

  - Lost term error
  - Request canceled by CLSDST
  - Error during start
  - Clean up or Notify
  - Z-net or cancel
- Send error.
  - z/OS cross-system coupling facility send failed.
  - An invalid data block (send error) is detected during a transmission (MSC environment only). The sender must handle the message that is in error. You can write the exit routine to check if the link is down or stopped at this time. DFS2140 with reason code 2146 indicates a send error.
  - An LU 6.2 session failed while sending an output message to an LU 6.2 program. The exit routine can only reroute or discard the message. The default action is to discard the message.
  - A send to an LU 6.2 program is rejected with a deallocate or with a send error. The exit routine can only ask IMS to reroute or discard the message. The default action is to discard the message.

  Restriction: When the exit routine discards a message from an LU 6.2 conversation because a send error occurred, the exit routine must not send an informational message to the originating LU 6.2 application. The informational message can be rejected for the same reason that the original message was rejected.

  If a send error occurs while sending a reply from an IMS local conversational transaction or a Fast Path transaction to an LU 6.2 program, this exit routine is not called. If the reply is from a remote transaction or a local nonconversational transaction, this exit routine is called.
- Receive error

  An input message error (receive error) is detected by the receiver of a message (MSC environment only). The following messages indicate a receive error: DFS064, DFS065, DFS076, DFS1959E, DFS2125, DFS2126, DFS2127, DFS2128, DFS2129, DFS2130, DFS2131, DFS2132, DFS2133, DFS2134, DFS2137, DFS2141, DFS2143, DFS2163, DFS2164, DFS2165, DFS2167, DFS2174, DFS2175, DFS2176, and DFS3470.
- A /DEQUEUE command with the specified *lterm*, *node*, *msname*, *luname* plus *tpname* and *tmember name* plus *tpipe name* keyword is entered. IMS calls the exit routine before processing each message on the queue.

Communicating with IMS

IMS uses the entry and exit registers, and the MSNB interface control block to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers in the provided save area. The registers contain the following information:

Register	Contents
1	Address of Message Control/Error exit interface block, MSNB.
13	Address of save area. The exit routine must not change the first 3 words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers. The contents of the interface block pointed to by register 1 can be different.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Rerouting messages

Given certain conditions, the Message Control/Error exit routine enables you to reroute transactions, responses, and message switches that are in error.

The Message Control/Error exit routine enables you to reroute transactions, responses, and message switches that are in error. When you reroute a message to a different destination, that destination must be a local or remote transaction, a local or remote LTERM, an LU 6.2 destination, or OTMA *tmember* and *tpipename*. The new destination must be capable of processing the message.

Restriction: You cannot reroute a message to a CPI-C driven application program.

An LU 6.2 destination is the LU 6.2 application program and is always defined with the LU name, plus the TP name.

A message that is rerouted to a transaction (conversational or nonconversational) can include the interface block if your exit routine sets the MSX2QBK bit in the MSXFLG2 field of the MSNB interface block. If this bit is on, a new message (with the interface block included) is built and enqueued to the new destination. If this bit is off, the original message is enqueued.

The format of the message depends on the message type and the new destination type as shown in the following table. Each destination type is discussed in the topic following the figure.

Table 78. Rerouting messages to new destinations

Message type	New destination	Message format (if MSX2QBK is turned on)
1. Conversational	Conversational transaction	SPA + interface block + message
2. Conversational	Nonconversational transaction	Interface block + unpacked SPA + message
3. Nonconversational (but not message switch)	Nonconversational transaction	Interface block + message
4. Message switch	Nonconversational transaction	Interface block + message
5. Message switch	LTERM	Original message
6. All types	Luname, tpname	
7. OTMA	Transaction, lterm, luname + tpname, or OTMA member name + tpipe name	Interface block + message if the new destination is transaction or lterm, the message format rules for message types 1 through 5 are applicable.

Recommendation: If a message must be rerouted, reroute it to a local nonconversational transaction to avoid further error. This nonconversational transaction is a special-purpose error processing transaction and can process all messages that are rerouted to it.

Attention: During the rerouting process, the original message is dequeued first, and then the newly built message is enqueued to the new destination. If a system failure occurs between the dequeue and enqueue processing, the message can be lost.

Subsections:

Rerouting to a conversational transaction

When a conversational message is rerouted to another conversational transaction, the scratchpad area (SPA) is the first segment, and the interface block is the next segment (if your exit routine sets the MSX2QBK bit). If you reroute a conversational transaction to a different conversational transaction, make sure that both transactions have the same SPA size.

Rerouting to a nonconversational transaction

When the new destination is a nonconversational transaction, the interface block is the first segment of the rerouted message (if your exit routine sets the MSX2QBK bit).

If the message is conversational, the segment following the interface block is the unpacked SPA and should be treated as a data segment by the new destination's application program. If the message is conversational or is in response mode (or both), it is the user's responsibility to end the conversation and take the input terminal out of response mode. One of the following can be done to end the conversation or take the terminal out of response mode:

- Enter the `/EXIT` command from the input terminal, if the keyboard is not locked.
- If the input terminal is a static terminal, from the MTO or system console of the input system, enter:
  - `/DISPLAY CONVERSATION HELD NODE nodename`
    or
    `/DISPLAY CONVERSATION BUSY NODE nodename`

    (to determine the conversation ID)
  - `/STOP NODE nodename`
  - `/EXIT CONVERSATION conversation id NODE nodename`
  - `/START NODE nodename`

    (if appropriate)

  These commands can also be issued from an AOI program.
- If the input terminal was dynamically created using the Extended Terminal Option (ETO) feature, from the MTO or system console of the input system, enter:
  - `/DISPLAY CONVERSATION HELD USER username` or
    `/DISPLAY CONVERSATION BUSY USER username`

    (to determine the conversation ID)
  - `/STOP USER username`
  - `/EXIT CONVERSATION conversation id USER username`
  - `/START USER username`

    (if appropriate)

  These commands can be issued from an AOI program.

Related Reading: For more information on these commands, see *IMS Version 15.2 Commands, Volume 1: IMS Commands A-M*.

Rerouting to an LTERM

When the new destination for a message is an LTERM and a message is rerouted from one physical terminal type to another, IMS rejects the message and issues an error message (such as DFS2078) if the new destination cannot handle the data.

Related Reading: For more information, see *IMS Version 15.2 Messages and Codes, Volume 1: DFS Messages*.

Related reference

"Message Control/Error Exit Interface Block (MSNB)" on page 212
The interface block for the Message Control/Error exit routine contains all of the information about the message, including contents of key fields as they appear on entry and exit. The exit fields are used to return information to IMS.

Message Control/Error Exit Interface Block (MSNB)

The interface block for the Message Control/Error exit routine contains all of the information about the message, including contents of key fields as they appear on entry and exit. The exit fields are used to return information to IMS.

The entry flag (MSNFLG1) indicates the reason the exit routine is called, and the exit flag (MSXFLG1) determines what action will be performed when control is returned to IMS. MSNBSEG1 points to the first segment of the message. If the segment is a SPA, IMS unpacks it before passing control to the exit routine. The exit routine can place any information that it needs into the user work area (MSNBUSRA); IMS does not disturb the contents of this work area.

The Message Control/Error exit routine can only modify seven fields: MSNBRTPG, MSNBRTPN, MSNBDEST, MSNBRINF, MSNBUSRA, MSXFLG1, and MSXFLG2. All other fields are read-only. If the exit routine modifies MSNBDEST, it must modify MSNBRINF. If the exit routine modifies MSNBRTPG and MSNBRTPN, it must modify MSNBRINF. In addition, the exit routine can modify MSXFLG2 if the exit routine modifies MSNBDEST and MSNBRINF, or MSNBRTPG, MSNBRTPN and MSNBDEST.

Subsections:

Contents of interface block on entry

The following table shows the contents of key fields in the Message Control/Error exit interface block as they appear on entry.

Table 79. Key fields of interface block on entry

Byte	Field name	Contents
X'C'	MSNFLG1	**Entry Flag** **Meaning** **X'80'** MSC link start **X'40'** MSC link termination **X'20'** Send error detected **X'10'** Receive error detected **X'08'** /DEQUEUE command entered **X’04’** /DEQUEUE command called CONU0 before exit **X'02'** DFS message send error detected **X’01’** Late response message to a synchronous program switch request
X'D'	MSNFLG2	**Entry Flag** **Meaning** **X'80'** Message prefix error detected **X'40'** Invalid data block detected **X'20'** LU 6.2 session failed or send action was rejected **X'04'** z/OS cross-system coupling facility send action failed
X'E'	MSNFLG3	**Entry Flag** **Meaning** **X'80'** DEQUEUE NODE command entered **X'40'** DEQUEUE LTERM command entered **X'20'** DEQUEUE MSNAME command entered **X'10'** DEQUEUE LUNAME TPNAME command entered **X'08'** DEQUEUE TMEMBER TPIPE name entered

Table 79. Key fields of interface block on entry (continued)

Byte	Field name	Contents
X'F'	MSNFLG4	**Entry Flag** **Meaning** **X'80'** Message is a transaction **X'40'** Message is a message switch **X'20'** Message is a response **X'10'** SPA in the message **X'08'** Response mode message **X'04'** Conversation starting **X'02'** Message switch from DFSAPPC **X'01'** Message from APPC type message
X'26'	MSNBOSID	Source SYSID (if MSC)
X'28'	MSNBDSID	Destination SYSID (if MSC)
X'2A'	MSNBMGID	Error message number (if receive error)
X'2C'	MSNBORGN	Message origin source name [1]
X'5C'	MSNBDSNM	Final destination of message
X'88'	MSNBRTPG	Length of TP name from /DEQ LU name TP name command
X'8A'	MSNBRMEM	Rerouted destination member name for OTMA or tmember name of /DEQ tmem/tpipe
X'8A'	MSNBRTPN	TP name from /DEQ LU name TP name command
X'CA'	MSNBDEST	• Node if /DEQ node command • LTERM if /DEQ lterm command • MSNAME if /DEQ msname command • LU name TP name if /DEQ luname tpname command
X'14E'	MSNBUSRA	User work area

Note: [1] In an LU 6.2 conversation, when the outbound message is re-enqueued across restart, the message origin source name (MSNBORGN) is blank.

Contents of interface block on exit

The following table shows the contents of key fields in the Message Control/Error exit interface block as they appear on exit. The exit routine uses these fields to return information to IMS.

Table 80. Key fields of interface block on exit

Byte	Field name	Contents
X'84'	MSXFLG1	**Exit Flag** **Meaning** **X'00'** No message is involved. (Perform the default action, which is the same action as in the prior release.) You can modify the exit routine to perform: • Initialization processing (including external IMS System Services) at link start • Clean-up processing at link termination **X'80'** Reroute the message to a different local or remote transaction, a local or remote LTERM, or an LU 6.2 destination. The exit routine must provide the new destination name in the MSNBDEST field, and set MSNBRINF to indicate an LTERM, a transaction, or an LU 6.2 destination. **X'60'** Perform actions of both X'20' and X'40'. **X'40'** Discard the message or proceed with the /DEQUEUE command. **X'30'** Perform actions of both X'10' and X'20'. **X'20'** If the exit routine selects this action, IMS sends an informational message: • If the /DEQUEUE command was entered, IMS sends DFS2185 to the entering terminal. • If IMS detected a receive error, IMS sends DFS2184 to the current MTO or input terminal. • If IMS detected a send error, IMS sends DFS2184 to the current MTO. If this action is selected by default and not by the exit routine, IMS sends an informational message: • On a send error, IMS sends DFS2140. • On a receive error, IMS sends the message number in the MSNBMGID field. This exit flag can be specified only in combination with exit flag X'10' or X'40'. **X'10'** Suppress the /DEQUEUE command. The /DEQUEUE PURGE operation is terminated if the exit routine requests to suppress the command.

Table 80. Key fields of interface block on exit (continued)

Byte	Field name	Contents
X'85'	MSXDFT1	**Exit Flag** **Meaning** **X'00'** No message involved (link start or link termination) or the default action. **X'80'** Reroute message to a different destination. **X'40'** Discard the message or proceed with the /DEQUEUE command. **X'20'** Send error message to current MTO or input terminal. **X'10'** Suppress the /DEQUEUE command.
X'86'	MSXFLG2	**Exit Flag** **Meaning** **X'80'** MSX2QBK field; include interface block in the message when rerouting to a different destination.
X'88'	MSNBRTPG	Length of rerouted TP name.
X'8A'	MSNBRMEM	Rerouted destination member name for OTMA or tmember name of /DEQ tmem/tpipe
X'8A'	MSNBRTPN	Rerouted TP name.
X'CA'	MSNBDEST	Destination name of local or remote transaction or local or remote LTERM, or reroute LU name or reroute netid.luname (left-justified, padded with blanks) if reroute the message.
X'105'	MSNBRINF	**Exit Flag** **Meaning** **X'80'** Destination is a transaction. **X'40'** Destination is an LTERM. **X'20'** Destination is a dynamic local LTERM. **X'10'** Destination of LU name plus TP name. **X'08'** Destination of OTMA member plus tpipe. **X’04’** Destination is the OTMA tmember and tpipe specified in the descriptor for a late response message to a synchronous program switch request.

Table 80. Key fields of interface block on exit (continued)

Byte	Field name	Contents
X'107'	MSNBRFL1	**Exit Flag** **Meaning** **X'80'** Destination is a local transaction. **X'40'** Destination is a remote transaction. **X'20'** Destination is a remote LTERM.
X'10D'	MSMFLG1	**Exit Flag** **Meaning** **X'80'** Next segment is a SPA. **X'40'** The two-byte SID was provided in the MSC extension.
X'12A'	MSNBMSG	Message area when error encountered in the interface module.
X'14E'	MSNBUSRA	User work area.

Logging the interface block

Two copies of the interface block are added to the existing X'6701' log record. The first copy is labeled "MSNB" and represents the interface block before IMS calls the Message Control/Error exit routine with the log record ID of CMEA. The second copy is labeled "USR MSNB" and represents the interface block after IMS calls the exit routine with the log record ID of CMEB. The X'6701' log record can be logged for informational reasons or to indicate an error in preparing to call the exit routine, or in performing the action(s) requested by the exit routine. The trace ID is CMEI. These log entries are forced entries for a send error, a receive error, and a /DEQUEUE command, regardless of any trace options that are specified. For link start and link termination, the interface block is only logged if the trace option is in effect on the link or node involved.

Related Reading: For more information on this log record, see *IMS Version 15.2 Diagnosis*.

Valid flags and default actions

IMS performs the default actions if the exit routine returns control to IMS without modifying the exit flag field, if the exit routine requests an invalid exit flag, or if IMS encounters an error while trying to perform the action requested by the exit routine

Default actions are specified in the MSXDFT1 field. The exit flag field (MSXFLG1) is located in the interface block. If an invalid exit flag is requested, IMS sends error message DFS2184 to the current MTO, in addition to performing the default action.

The following table shows valid entry flags, exit flags, and default actions.

Table 81. Flags and default actions

Entry flag (MSNFLG1)	Valid exit flags (MSXFLG1)	Default action (MSXDFT1)
X'80'	X'00'	X'00'
X'40'	X'00'	X'00'

Table 81. Flags and default actions (continued)

Entry flag (MSNFLG1)	Valid exit flags (MSXFLG1)	Default action (MSXDFT1)
X'20'	X'00', X'40', X'60', X'80'	X'60' + stop MSNAME
X'10'	X'00', X'40', X'60', X'80'	X'60'
X'08'	X'00', X'10', X'30', X'40', X'80'	X'40'

Note: The default action for a send error (entry flag = X'20') includes STOP MSNAME. In addition, the default action for the DEQUEUE command is to proceed with the command. If you do not want these actions to take place, specify a different exit flag depending on the actions that you want to occur.

If any errors are encountered while IMS tries to perform the requested action, the action is ignored and the default action is performed. The MSNBMSG field of the interface block of the forced 6701 CMEI log record will contain one of the following brief descriptions that describe the error encountered, if applicable:

- No storage for message buffer
- Invalid destination for reroute
- Cannot reroute MSG switch to CONV
- Error while building rerouted MSG
- Reroute destination not found
- Cannot reroute CONV MSG to LTERM
- Cannot reroute non-CONV MSG to CONV

Related reference

"Message Control/Error Exit Interface Block (MSNB)" on page 212
The interface block for the Message Control/Error exit routine contains all of the information about the message, including contents of key fields as they appear on entry and exit. The exit fields are used to return information to IMS.

Message Switching (Input) edit routine (DFSCNTE0)

This Message Switching (Input) edit routine (DFSCNTE0) is called when a message is entered from a terminal with EDIT=(YES,...) in the NAME macro to another terminal

This topic contains Product-sensitive Programming Interface information.

This topic describes the Message Switching (Input) Edit routine. Information about using a sample routine is provided at the end of this topic.

Subsections:

About this routine

A facility similar to the Transaction Code (Input) Edit is provided for message switching. The optional user-written routine, whose CSECT and load module name must be DFSCNTE0, is loaded as stand-alone modules during IMS initialization. Only one Message Switching edit routine can be specified for an IMS online control program. This routine is specified for inclusion with the online control program by specifying EDIT=(YES,...) in one or more NAME macros during system definition. It is not called when the message is inserted using a program-to-program switch.

The Message Switching (Input) edit routine does not support terminals that are defined dynamically using the Extended Terminal Option (ETO) feature.

Related Reading: For more information on ETO, see *IMS Version 15.2 Communications and Connections*.

The following table shows the attributes of the Message Switching (Input) edit exit routine.

Table 82. Message switching (input) edit exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCNTE0.
Including the routine	This exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. Manually link this exit with DFSCSI00 to use callable services. No additional linking is required To use IMS callable services.
Sample routine location	IMS.ADFSSMPL (member name DFSCNTE0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of registers on entry

On entry, the edit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	The buffer location of the input message segment after translation to EBCDIC and after IMS Basic Editing. The first two bytes of the buffer contain a binary message length. The third byte of the buffer is binary zeros. The binary count includes the 4-byte prefix. The fifth byte contains the first byte of message text.
7	Address of CTB.
9	Address of CLB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

Use the message segment in the buffer addressed by register 1 as input to the edit routine.

The edit routine must place the text of the edited message segment to be returned to IMS in the buffer addressed by register 1. If the input was processed by the IMS Basic Edit, this buffer is always 10 bytes greater than the 2-byte binary count at the beginning of the message segment. The length of the message segment can be expanded or reduced to any desired size. The format of the edited message segment in the buffer on return to IMS must be two bytes of binary count (LL), two bytes of binary zeros (ZZ), and edited text. The second two bytes (ZZ) should not be changed or edited. The LLZZ field is the first four bytes of the message segment.

Contents of registers on exit

Before returning to IMS, the edit routine must restore all registers except register 15, which must contain one of the following return codes.

Return code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled and the user message identified by register 1 is sent to the terminal.

Register 1 contains the message number if register 15 contains a return code of 12; otherwise it is ignored. Any other value causes the message to be canceled and the terminal operator to be notified.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Using the sample message switching edit routine (DFSCNTE0)

The edit routine can be used to identify, in the text of the message to the output terminal, the logical terminal name from which the message was entered and the message number.

In the example, the input logical terminal name is used. This name is found in the Communication Name Table (CNT), which is the IMS control block for the input logical terminal. The CNT is addressed by a field called CTBCNTPT in the Communication Terminal Block. The field in the CNT containing the logical terminal name is called CNTNAME. Control blocks are defined in *IMS Version 15.2 Diagnosis*.

NDMX: Non-Discardable Messages user exit (DFSNDMX0 and other NDMX exits)

Use the Non-Discardable Messages exit routine to tell IMS what to do with the input message associated with an abended application program.

If IMS does not call the Non-Discardable Messages exit routine, IMS arbitrarily discards messages from the system and issues message DFS555I.

Subsections:

About this routine

The Non-Discardable Messages exit routine receives control when an IMS application abends with an input message in process.

Note: Non-message-driven BMP regions do not receive input messages, but you can use this routine to specify what IMS does when a program abends in these regions. Details are available throughout this topic.

Attributes of the Non-Discardable Messages exit routine are as follows:

Table 83. Non-discardable messages exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You can name this exit routine DFSNDMX0 and link it into a library that is included in the STEPLIB concatenation. If DFSNDMX0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines in the DFSDFxxx member are loaded. DFSNDMX0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDF*xxx* member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.
Binding	This exit routine must be reentrant. It runs in non-cross-memory mode.

Including the routine	If you write your own exit routine and plan to use IMS callable services, you must manually link edit the routine withDFSCSI00, and you must link the routine with IMS.SDFSRESL. The following example shows the required bind JCL statements.
	
	`INCLUDE LOAD(DFSNDMX0)`
	`INCLUDE LOAD(DFSCSI00)`
	`ENTRY   DFSNDMX0`
	`NAME    DFSNDMX0(R)`
	
	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. To use a single exit routine that is named DFSNDMX0, no additional steps are needed. By default, the exit is activated for MPR, JMP, and message-driven BMP region types; IFP and non-message-driven BMP regions are not activated. To use multiple exit routines, specify **EXITDEF=(TYPE=NDMX,EXIT=(exit_names))** in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
	
	To activate support for message-driven IFP regions or non-message-driven BMP regions, specify the additional parameter **NDMX_CALLED_FOR()** in the USER_EXITS section of the DFSDFxx member and include the following options:
	
	• To activate support for IFP regions, specify **NDMX_CALLED_FOR(IFP(Y))**.
	• To activate support for only non-message-driven BMP regions, specify **NDMX_CALLED_FOR(BMP(NMD_ONLY))**. This activates support for non-message-driven BMP regions and **deactivates** default activation for message-driven BMP regions.
	• To activate support for both message-driven and non-message-driven BMP regions, specify **NDMX_CALLED_FOR(BMP(ALL))**. As a shorter alternative, you can use NDMX_CALLED_FOR(BMP(A)).
	
	To learn more, see USER_EXITS section of the DFSDFxxx member (System Definition).

Table 83. Non-discardable messages exit routine attributes (continued)

Attribute	Description
IMS callable services	To use callable services with this routine, examine the value of the SXPLATOK field in the IMS standard user exit parameter list to see whether a callable services token is passed to the routine. • If SXPLATOK is zero, you cannot use callable services with this routine. • If SXPLATOK is nonzero, the callable services token is included, and you can use callable services. You can use the 256-byte work area that is addressed by SXPLAWRK in the standard user exit parameter list to call DFSCSIF0.
Sample routine location	IMS.ADFSSMPL (member name DFSNDMX0). The mapping of the NDM interface block is available from the IMS library IMS.ADFSMAC (member name DFSNDM).

Processing options

The following processing options are valid for DFSNDMX0. If you request an option that is not valid, IMS ignores your request and continues normal processing (the default option).

Continue normal processing

Continue normal processing is the default option. Request this option by setting register 15 to zero before returning to IMS. IMS proceeds as if this exit routine had not been called.

Depending on the type of application abend that initiated the exit routine, IMS might delete the message, issue a DFS555I message to the originating terminal and master terminal, and issue a DFS554A message to the master terminal.

Delete the input message from the system

Request this option by setting register 15 to 4 before returning to IMS. If you request this option, IMS performs the following actions:

1. Issues a DFS555I message to the originating terminal (if possible) and to the master terminal
2. Deletes the input message from the system
3. Issues a DFS554A message to the master terminal

Queue the message to the suspend queue

Request this option by setting register 15 to 8 before returning to IMS. If you request this option, IMS queues the input message to the suspend queue of the transaction that was being processed when the application abended. IMS suspends the transaction, and might issue a DFS554A message to the master terminal based on the abend type that occurred.

This option is not applicable to IFP and non-message-driven BMP regions.

Requeue the input message to the original transaction

Request this option by setting register 15 to 12 before returning to IMS. If you request this option, IMS queues the input message to the normal processing queue of the transaction that was being processed when the application abended. IMS USTOPs the transaction unless directed to do otherwise by the contents of NDMTRNST, and might issue a DFS554A message to the master terminal depending on the abend type that occurred.

This option is not applicable to IFP and non-message-driven BMP regions.

Queue the message to an alternate destination

Request this option by setting register 15 to 16 before returning to IMS and placing a valid destination name in the NDMDEST field of the NDM interface block. The following table shows the valid destination types and how to specify them in NDMDEST.

This option is not applicable to IFP and non-message-driven BMP regions.

Table 84. Valid alternative destinations

Alternative destination	NDMDEST value
LTERM	Specify a local, remote, or ETO LTERM, using the LTERM name or ETO user descriptor name.
OTMA	Specify the OTMA TPIPE name, or a name that is meaningful to the OTMA exit routines.
LU 6.2	Specify a local LU 6.2 device descriptor. The LU 6.2 device must be on the local IMS subsystem.
Transaction	Specify a local or remote transaction code. The following transaction types are **not** valid destinations: • Fast Path exclusive transaction. • Conversational transaction. • SAA communications-driven transaction (that is, a CPI-C driven transaction). If you specify an invalid transaction type, IMS ignores the request and continues normal processing.

If NDMDEST contains an invalid destination, such as zeros or blanks, IMS ignores the request to change the destination and continues normal processing.

If NDMDEST contains a destination that is unknown to IMS, processing depends on whether OTMA, and ETO or shared queues are active.

With OTMA, and ETO or shared queues active
IMS starts the OTMA exit routines before it starts the Destination Creation exit routine (DFSINSX0).

Without OTMA, ETO, or shared queues
IMS ignores the request and continues normal processing.

When IMS requeues the input message to a valid destination, IMS completes the message processing as follows:

1. Issues a DFS550I message (succeeded version) to the master terminal
2. Issues a DFS555I message to the originating terminal (if possible) and to the master terminal
3. Deletes the input message from the abended transaction
4. Issues a DFS554A message to the master terminal

Restrictions

Not all destinations are valid alternatives for input messages. You can use this exit routine to requeue messages to alternative destinations.

The following processing options are not supported for a message-driven Fast Path region:

- Queue the message to the suspend queue.
- Requeue the input message to the original transaction.
- Queue the message to an alternative destination that is named in the NDMDEST field in the NDM interface block.

Communicating with IMS

This exit routine uses a parameter list, entry and exit registers, and the Non-Discardable Messages interface block (NDM) to communicate with IMS.

Contents of registers on entry

At entry, the exit routine must save all registers that use the provided save area. The registers contain the following items:

Register	Content
1	Address of the “IMS standard user exit parameter list” on page 5.
13	Address of a single standard z/OS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard user exit parameter list

This exit routine uses the Version 6 standard exit parameter list. The address of the work area passed to this exit routine in SXPLAWRK can be different each time that this exit routine is called.

If your NDMX user exit can be called in an enhanced user exit environment, more user exit routines might be called after your routine. When your user exit routine finds a transaction to act upon, it can set SXPL_CALLNXTN in the byte that SXPLCNXT points to, which tells IMS to not call more exit routines.

NDM interface block

The following table shows the contents of the NDM interface block. The address of this parameter list is in the standard user exit parameter list (field name SXPLFSPL). The mapping of the NDM interface block is available from the IMS library IMS.ADFSMAC (member name DFSNDM).

Table 85. NDM interface block

Field	Offset	Length	Content
NDMEYE	X'00'	4	NDM eye catcher.
NDMTRAN	X'04'	8	Transaction that the application was processing when it abended. This transaction is associated with the input message pointed to by NDMMSGA. For non-message-driven BMP regions, the NDMTRAN code is always 0 (zero).
NDMPSBN	X'0C'	8	PSB associated with the application that abended.
NDMUSID	X'14'	8	User ID.
NDMGRPNM	X'1C'	8	Group name.
NDMUSIDI	X'24'	1	Character flag for contents of user ID field NDMUSID: **Character** **Meaning** **U** User ID **L** LTERM **P** PSB name **O** Other name

Table 85. NDM interface block (continued)

Field	Offset	Length	Content
NDMSRCFL	X'25'	1	A flag that indicates the origin of the input message. This flag is set with one of the following values: **Value** **Meaning** **0** NDMLTERM The source of the input message is an LTERM. Subsequent fields contain information about the LTERM. **1** NDMOTMA The source of the input message is OTMA. Subsequent fields contain information about the OTMA source. **2** NDMLU62 The source of the input message is an LU 6.2 device. Subsequent fields contain information about the LU 6.2 device. **3** NDMNMSG Indicates that this is a non-message region and that there is no input message.
NDMSRCIN	X'26'	1	Start of source description.
NDMLTERM	X'26'	8	Name of the originating LTERM if NDMSRCFL is set to NDMLTERM (value 0).
NDMNOMSG	X'26'	8	Non-message-driven regions set the value NDMNOMSG, which signifies a non-message-driven region with no source input message available, and is indicated by 8 bytes of blanks.
NDMTPIPE	X'26'	8	OTMA TPIPE name if NDMSRCFL is set to NDMOTMA (value 1).
NDMMEM	X'2E'	16	OTMA member name.
NDMTPSYN	X'3E'	1	OTMA TPIPE synchronization level.
NDMMGSYN	X'3F'	1	OTMA message synchronization level.
NDMLUNM	X'26'	8	LU name if NDMSRCFL is set to NDMLU62 (value 2).
NDMNWID	X'2E'	8	Network identifier.
NDMSIDE	X'36'	8	APPC side information name.
NDMMODE	X'3E'	8	VTAM mode table name.
NDMTPNML	X'46'	2	Length of TP name contained in NDMTPNM.
NDMTPNM	X'48'	64	TP name.

Table 85. NDM interface block (continued)

Field	Offset	Length	Content
NDMCONV	X'88'	1	APPC conversation type.
NDMSYNC	X'89'	1	APPC synchronization level.
	X'8A'	17	Reserved.
NDMRGTYP	X'9B'	1	IMS Dependent Region type: **Value Description** **x'88'** Java Message Processing Region (JMP). **x'80**' Message Processing Region (MPR). **x'50'** Message-driven Fast Path Region (IFP). **x'40'** Message-driven Batch Message Processing Region (BMP).
NDMABEND	X'9C'	4	Abend code in system format `00sssuuu`, where: **sss** z/OS system abend code. **uuu** IMS user abend code.
NDMTSLCL	X'A0'	8	The local timestamp of the arrival of the input message in the system. NDMTSLCL contains the two fields NDMDLCL and NDMTLCL.
NDMDLCL	X'A0'	4	The local date that the message arrived in the system. The date format is `YYYYDDDf`, where: **YYYY** Year. **DDD** Julian day. **f** X'F'.
NDMTLCL	X'A4'	4	The local time that the message arrived in the system. The time format is `HHMMSSTf`, where: **HH** Hour. **MM** Minutes. **SS** Seconds. **T** Tenths of the second. **f** X'F'.

Table 85. NDM interface block (continued)

Field	Offset	Length	Content
NDMTSUTC	X'A8'	12	The Coordinated Universal Time (UTC) timestamp of the arrival of the input message in the system. The timestamp format is as follows: **Year/day** YYYYDDDf **Time** HHMMSSTHmiju **Offset** Aqq$ The timestamp fields include the following items: **YYYY** Year. **DDD** Julian day. **f** X'F'. **HH** Hour. **MM** Minutes. **SS** Seconds. **T** Tenths of the second. **H** Hundredths of the second. **m** Milliseconds. **i** Tenths of a millisecond. **j** Hundredths of a millisecond. **u** Microseconds. **A** Attribute of the time value. **qq** Quarter-hours of offset from Coordinated Universal Time (UTC). **$** Decimal sign for the offset, either positive (X'C') or negative (X'D').

Table 85. NDM interface block (continued)

Field	Offset	Length	Content
NDMSPAA	X'B4'	4	Address of the SPA if the transaction in NDMTRAN is a conversational transaction. Otherwise, this field contains zeros. If the SPA is present, the format is as follows: `LL ZZZZ transaction_code data` **LL** Two-byte length field that includes the length of LLZZZZ. **ZZZZ** Four-byte field that always contains zeros. **transaction_code** Eight-byte field that contains the transaction code for the conversation or blanks. **data** SPA user data.
NDMMSGA	X'B8'	4	Contains the address of the input message if this field is nonzero. If this field is zero, there is no message segment, and can be an SPA segment only. The message format is as follows: `LL ZZ message-segment` **LL** Two-byte length field that includes the length of LLZZ. **ZZ** Two-byte field that always contains zeros, except for the last message segment, which contains X'FFFF'. **message-segment input** message segment For a single-segment message, the pattern is: LL=NDMMSGL and ZZ=X'FFFF'. For a multi-segment message, the pattern is: • NDMMSGA=address of first segment. • NDMMSGA+LL=address of second segment. • NDMMSGA+LL+LL= address of third segment.
NDMMSGL	X'BC'	4	Total length of input message
	X'C0'	20	Reserved
NDMABRSN	X'D4'	4	Abend reason code if available.

Table 85. NDM interface block (continued)

Field	Offset	Length	Content
NDMTRNST	X'D8'	4	Transaction status flag. DFSNDMX0 can set this field to any one of the following values. IMS examines this field on return from DFSNDMX0. **Value** **Description** **1** Do not (U)STOP the abended transaction and do not STOP the abended program. **2** Do not send the DFS555I message. **3** Do not (U)STOP the abended transaction, do not STOP the abended program, and do not send the DFS555I message. **4** Allow messages to continue to queue for the transaction, but do not allow the transaction to continue to be scheduled. This is equivalent to the PSTOP TRAN command for this transaction only. The PSB and application program are not affected. PSTOP is not valid for fast-path exclusive transaction and is ignored. **5** Allow the transaction to continue to schedule, but do not allow messages to continue to queue for the transaction. This is equivalent to the PURGE TRAN command for this transaction only. The PSB and application program are not affected. **6** Stop the transaction. This is equivalent to the STOP TRAN command. The PSB and application program are not affected.

Table 85. NDM interface block (continued)

Field	Offset	Length	Content
NDMTRNST (continued)	X'D8'	4	**7** Start the transaction. This is equivalent to the START TRAN command. The PSB and application program are not affected. For non-message-driven BMP regions, only codes 1 and 3 apply. For IFP regions, the allowed values that can be specified for NDMTRNST can be any number 1 through 7. However, if **NDMTRNST=4 (PSTOP)** is specified and the transaction is defined as Fast-Path, exclusive is ignored and the transaction status is set to STOP. This behavior is consistent with the way IMS handles a PSTOP command that is issued against a Fast-Path exclusive transaction. The recommended value for NDMTRNST is 1, which enables the Do not stop Transaction and Program. For both return codes 0 and 4, the Transaction and Program is not stopped. For non-retriable abend situations in IFP regions, the recommended return code is 4. For retriable abends, you cannot stop the program, and the recommended return code is 0. Any other return code is treated as return code 0. The user selected return code is printed in the **'67D0'** log record for diagnostics purpose.
NDMDEST	X'DC'	8	Name of the alternative destination to which the input message is to be queued. IMS examines this field only if you pass return code 16 in register 15. Otherwise, IMS ignores this field.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return code	Meaning
0	Continue normal processing.
4	Delete the input message from the system. This return code is not applicable for non-message-driven BMP regions.
8	Queue the input message to the suspend queue. This return code is not applicable to IFP or non-message-driven BMP regions.
12	Requeue the input message to the original transaction. This return code is not applicable to IFP or non-message-driven BMP regions.
16	Queue the message to an alternative destination that is named in the NDMDEST field in the NDM interface block. This return code is not applicable to IFP or non-message-driven BMP regions.

For return codes 0 and 4, the Transaction and Program are not stopped. For non-retriable abend situations in IFP regions, the recommended return code is 4. For retriable abends, you cannot stop the program, and the recommended return code is 0. Any other return code is treated as return code 0. The user selected return code is printed in the **'67D0'** log record for diagnostics purpose.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

"OTMA User Data Formatting exit routine (DFSYDRU0)" on page 241
The OTMA User Data Formatting exit routine can determine and change the final destination of OTMA messages. The DFSYDRU0 exit routine can also format the User Prefix section of an OTMA asynchronous output message.

"OTMA Input/Output Edit user exit (DFSYIOE0 and other OTMAIOED type exits)" on page 236
You can use the OTMA Input/Output Edit user exit to modify or cancel IMS Open Transaction Manager Access (OTMA) input and output messages. You can also use this user exit to format the User Prefix section of an OTMA input or output message.

"OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits)" on page 231
The OTMA Destination Resolution user exit determines whether an asynchronous output message needs to be routed to an OTMA destination or a non-OTMA destination. If the message should be routed to an OTMA destination, the user exit can determine the final OTMA destination client or Tpipe.

"Destination Creation exit routine (DFSINSX0)" on page 147
The Destination Creation exit routine creates an LTERM or a transaction when a destination for a message does not exist.

OTMA Destination Resolution user exit (DFSYPRX0 and other OTMAYPRX type exits)

The OTMA Destination Resolution user exit determines whether an asynchronous output message needs to be routed to an OTMA destination or a non-OTMA destination. If the message should be routed to an OTMA destination, the user exit can determine the final OTMA destination client or Tpipe.

You can use OTMA destination descriptors as an alternative to coding an OTMAYPRX user exit.

Subsections:

About this routine

The following rules apply for this user exit:

- This routine is optional, and can be written so that IMS data is not prerouted.
- If the destination name is an IMS scheduler message block (SMB) name, this routine cannot change it.
- Transaction output can be directed to an OTMA client, even if the transaction originates from a non-OTMA source.

- Transaction output can be directed to a non-OTMA destination, even if the transaction originates from an OTMA client.
- In an IMS subsystem, only one OTMA Destination Resolution user exit is allowed.

Important: Within a shared-queues group, ensure that the OTMAYPRX user exit is the same for both front-end and back-end IMS systems. If these exit routines differ on 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.

If multiple user exits routines are used, ensure the OTMARTUX user exit routines are defined in the same order on front-end and back-end IMS systems.

The following table shows the attributes of the OTMA Destination Resolution user exit.

Table 86. OTMA Destination Resolution user exit attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You can name this exit routine DFSYPRX0 and link it into a library that is included in the STEPLIB concatenation. If DFSYPRX0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSYPRX0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.
Link editing	The OTMA Destination Resolution user exit must be reentrant. The OTMA Destination Resolution user exit must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. This exit routine is optional.
Including the routine	Add Including the routine section of the OTMA Destination Resolution user exit routine attributes that says the following: The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSYPRX0. If you use multiple exit routines, specify EXITDEF=(TYPE=OTMAYPRX,EXIT=(*exit_names*)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
IMS callable services	This user exit is eligible to use IMS callable services. To use callable services, examine the value of the SXPLATOK field in the IMS standard exit parameter list to determine if a callable services token was passed to the routine. If the value of the field is zero, no callable services are available. If the value is non-zero, examine the value of the SXPLAWRK field in the parameter list for the address to a 256-byte work area. Use the work area to issue calls to DFSCSIF0.
Sample routine location	IMS.ADFSSMPL (member name DFSYPRX0).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the user exit.

Contents of registers at entry

At entry, the user exit must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5
R13	Save area address (points to a single save area, not a save area chain)
R14	Return address
R15	Entry point address

Standard exit parameter list

This user exit uses the Version 6 standard exit parameter list. The address of the work area passed to this user exit in SXPLAWRK will be the same each time that this exit routine is called.

If your OTMAYPRX user exit can be called in an enhanced user exit environment, additional user exit routines can be called after your routine. When your user exit routine finds a message upon which to act, it can set SXPL_CALLNXTN in the byte SXPLCNXT points to. This tells IMS to not call additional exit routines.

Function specific parameter list on entry

The following table describes the contents of the OTMA Destination Resolution user exit parameter list. The address of this parameter list is in standard exit parameter list field SXPLFSPL.

Table 87. Contents of the OTMA Destination Resolution user exit parameter list

Offset (decimal)	Description
+0	Name of the originating LTERM or OTMA transaction pipe.
+8	Destination name.
+16	Transaction name or program name.

Table 87. Contents of the OTMA Destination Resolution user exit parameter list (continued)

Offset (decimal)	Description
+24	Flag byte: **Flag bits** **Description** **X'80'** An OTMA prefix exists. **X'20'** An OTMA message was submitted by an OTMA client with super member support. The OTMA state data pointed to by the input parameter list contains a 1-4 byte super member name at offset X'E' from the start of the state data. **X'10'** A DL/I ICAL call for synchronous program switch was issued. If the X'80' flag is also set, this flag indicates that an OTMA transaction initiated the ICAL call and the LTERM or tpipe name and input client member name in the exit parameter list are from the original OTMA transaction. **X'08'** The destination name matches an entry in the OTMA destination descriptor. The name is for an IMS Connect destination. **X'04'** The destination name matches an entry in the OTMA destination descriptor. The name is for an IBM MQ destination. **X'02'** The destination name matches an entry in the OTMA destination descriptor. The name is for a non-OTMA destination. **X'01'** If set, indicates that an ECB is passed for the exit to use. Do not cause delays from this exit because it will impact OTMA performance.
+25	Synchronization level.
+26	Reserved.
+27	A 1-byte field that indicates the version of the exit routine parameter list: **Flag bits** **Description** **X'80'** If set, indicates that at offset +88, the user exit parameter list includes the 4-byte address of the OTMA destination descriptor information. **X'40'** If set, indicates that at offset +96, the user exit parameter list includes the address of the OTMA security data prefix.
+28	User ID.
+36	Group name.
+44	Address of the PST block.
+48	Name of the originating OTMA client, if the message originated from an OTMA client; otherwise zeros.

Table 87. Contents of the OTMA Destination Resolution user exit parameter list (continued)

Offset (decimal)	Description
+64	Address of the input Message Control Information prefix section of the OTMA message. If this call is from an ICAL request for synchronous program switch, the message control information is generated by IMS. The information is not propagated from the original message prefix. However, the LTERM or TPIPE name and input client name are passed from the original OTMA message.
+68	Address of the input State Data prefix section of the OTMA message. Check the prefix flag in the Message Control Information section to determine the specific type of State Data section specified. If this call is from an ICAL request for synchronous program switch, the state data information is generated by IMS. The information is not propagated from the original message prefix. However, the correlator field, TMAMHCOR, is passed from the original OTMA state data. The LTERM or TPIPE name and input client name are also passed from the original OTMA message.
+72	Address of the input User Data prefix section of the OTMA message.
+76	Address of SCD control block.
+80	Address of the 16-byte client override name, if any, to be returned to IMS. This field is set by IMS at entry. It points to a 16-byte buffer area to which the OTMA client name is written, if one does not exist at entry. Do not alter this address. The OTMA client name is written when the transaction originates from a non-OTMA LTERM and is to be routed to an OTMA destination. If the transaction is invoked from an OTMA client and the OTMAMD initialization parameter is set to Y in the DFSPBxxx PROCLIB member, the client override name is accepted.
+84	Address of the 8-byte Tpipe override name, if any, to be used for OTMA output message queuing and transmission. If blank, this field is ignored. If specified, this field is omitted when the ALTPCB output is triggered by an IMS Connect commit-then-send send-receive call and the SENDALTP function is activated.
+88	Address of the OTMA destination descriptor. The address points to a location where the routing information that is defined in the descriptor for IBM MQ and IMS Connect is stored. If the destination name is for a non-OTMA destination, or if the destination name does not match any entry in the OTMA destination descriptor, this field is set to 0. For detailed information about IMS Connect destination routing, see the TMAMICON_DESCRIPTOR DSECT mapping. For detailed information about IBM MQ destination routing, see the TMAMMQS_DESCRIPTOR DSECT mapping.
+96	Address of the OTMA security data, which is available in the input message prefix. This is an entry parameter only.

Any other return code generates a DFS2370I message.

Contents of registers at exit

Before returning to IMS, the exit routine must restore all registers, except register 15, which must contain one of the following return codes:

Return code	Meaning
0	Input message came from OTMA, destination is same or different OTMA client. Or, input message did not come from OTMA, output is not OTMA
4	The message did not originally come from OTMA, but its destination is OTMA. **Note:** You must set the z/OS cross-system coupling facility member name of the OTMA client
8	The message came from OTMA, but the destination is not OTMA.
100	Use the routing information from the destination descriptor without modification. If the destination is IMS Connect or IBM MQ, the DFSYDRU0 exit routine is called and can modify the routing information. This return code is valid only when EXIT=Y is specified for the destination descriptor.

For the OTMAYPRX user exit, any other return code generates a DFS2370I message with the return code listed in hex. The hex equivalents for the return codes are:

0
X'00'

4
X'04'

8
X'08'

100
X'64'

Error conditions

An A1 status code is returned to the application program when the following errors occur:

- Incorrect 16-byte OTMA client override name is specified. The client name cannot contain all blanks or zeroes. If the client name is shorter than 16 bytes, it must be padded with blanks.
- Incorrect return code is specified for the exit.

Related concepts
OTMA destination descriptors (Communications and Connections)
Related reference
“OTMA User Data Formatting exit routine (DFSYDRU0)” on page 241
The OTMA User Data Formatting exit routine can determine and change the final destination of OTMA messages. The DFSYDRU0 exit routine can also format the User Prefix section of an OTMA asynchronous output message.

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

OTMA Input/Output Edit user exit (DFSYIOE0 and other OTMAIOED type exits)

You can use the OTMA Input/Output Edit user exit to modify or cancel IMS Open Transaction Manager Access (OTMA) input and output messages. You can also use this user exit to format the User Prefix section of an OTMA input or output message.

Subsections:

About this routine

This user exit can do the following for OTMA input and output messages:

- Modify the length or data of a message segment.

  IMS sends the modified message after it receives control from the user exit.
- Cancel a message segment.
- Cancel a message.

However, this user exit cannot be used for OTMA synchronous callout messages using DL/I ICAL calls.

If your OTMAIOED user exit can be called in an enhanced user exit environment, additional user exit routines might be called after your routine. When your user exit routine finds a transaction upon which to act, it can set SXPL_CALLNXTN in SXPL_FLGA. This tells IMS to not call additional exit routines.

Table 88. Canceling a message segment

Segment being canceled	IMS sends
First	The full OTMA message prefix, with null data.
Last	The last segment, with null data.
Other	Nothing. IMS does not send the message segment.

Table 89. Canceling a message

Segment being canceled	IMS sends
First	Nothing. IMS does not send the message, and returns a status code.
Other	The last segment, with null data. In the OTMA prefix, the "discard chain" flag is set.

The length of each message segment is limited to 32 KB. If a message segment exceeds this limit, IMS issues message DFS1294E, and processes the message as follows:

Segment being processed	IMS sends
First	The full OTMA message prefix, with null data.
Last	The last segment, with null data.
Other	Nothing. IMS does not send the message segment.

The following table shows the attributes of the OTMA Input/Output Edit user exit.

Table 90. OTMA input/output edit exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.

Table 90. OTMA input/output edit exit routine attributes (continued)

Attribute	Description
Naming convention	You can name this user exit DFSYIOE0 and link it into a library that is included in the STEPLIB concatenation. If DFSYIOE0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSYIOE0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more user exit modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.
Binding	The OTMA Input/Output Edit user exit must be reentrant. The OTMA Input/Output Edit user exit must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. This user exit is optional.
Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single user exit that is named DFSYIOE0. If you use multiple user exits, specify EXITDEF=(TYPE= OTMAIOED,EXIT=(*exit_names*)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
IMS callable services	This user exit is eligible to use IMS callable services. To use IMS callable services with this routine, examine the value of the SXPLATOK field in the “IMS standard user exit parameter list” on page 5 to see if a callable services token is available. If the value of SXPLATOK is zero, you cannot use callable services with this routine. If the value of SXPLATOK is non-zero, the callable services token is included, and you can use callable services. You can use the 256-byte work area addressed by SXPLAWRK in the standard user exit parameter list to call DFSCSIF0.
Sample routine location	IMS.ADFSSMPL (member name DFSYIOE0).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the user exit.

Contents of registers at entry

At entry, the user exit must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5
R13	Save area address
R14	Return address
R15	Entry point address

Standard exit parameter list

This user exit uses the “Version 6 standard exit parameter list” on page 6. The address of the work area passed to this user exit in SXPLAWRK will be the same each time that this exit routine is called.

If your OTMAIOED user exit can be called in an enhanced user exit environment, additional user exit routines might be called after your routine. When your user exit routine finds a transaction upon which to act, it can set SXPL_CALLNXTN in the byte that SXPLCNXT points to. This tells IMS to not call additional exit routines.

Function-specific parameter list on entry

The following are the contents of the OTMA Input/Output Edit user exit parameter list. The address of this parameter list is in standard exit parameter list field SXPLFSPL.

Table 91. OTMA Input/Output Edit user exit parameter list

Offset	Contents
+0	Input/output flag. Set to 0 for an input message segment; set to 4 for an output message segment.
+1	Segment-type flag. Set to 0 for the first message segment; set to 4 for any other message segment.
+2	**Flags** **Description** **X'40'** If set, indicates that at offset +52, the user exit parameter list includes the address of the OTMA security data prefix.
+4	Address of the message segment. The segment has the format LLZZDD: **LL** Total length (2 bytes) **ZZ** Flag (2 bytes). Z1 is reserved for IMS. The exit routine can change Z2. **DD** Message segment If the user exit modifies the message segment, it must also modify the LL with the new segment length. For null segments, LL must be set to 4 (2 bytes for LL and 2 bytes for ZZ). The user exit can increase any segment to a maximum of 256 bytes. The overall message, however, cannot exceed 32767 bytes (including the LL and ZZ fields). If a segment exceeds the 256-byte limit, IMS truncates it and issues message DFS1967.
+8	Address of the transaction code.
+12	Address of the OTMA transaction pipe name.
+16	Address of the z/OS cross-system coupling facility member name.
+20	Address of the user ID.
+24	Address of the OTMA user table, if any.
+28	Address of the message control region, available from input/output message prefix. This is an entry parameter only.
+32	Address of state data, available from input/output message prefix. This is an entry parameter only.

Table 91. OTMA Input/Output Edit user exit parameter list (continued)

Offset	Contents
+36	Address of user data, available from input/output message prefix. This area can be used to return modified user data, but the length of user data cannot be changed. The format of the user data is: **0-1** Length of the user data that follows (including this length field). This user exit cannot change the length of user data. **2** User data.
+40	Address of the output parameter list. The output parameter list is used to return information to IMS and is defined as follows: **+00** 8-byte LTERM override. This field is used to override the destination override specified in the state data. **+08** 8-byte map name override. This field is used to override the map name specified in the state data. **+16** **Flag** **Description** **X'80'** Wait for write for CM1 Fast Path transaction **X'00'** Request check write for CM1 Fast Path transaction. **+17** Reserved.
+44	Address of the SCD.
+52	Address of security data, which is available in the input and output OTMA message prefix. This is an entry parameter only.

Contents of registers at exit

Before returning to IMS, the user exit must restore all registers, except register 15, which must contain one of the following return codes:

Return code	Meaning
0	Processing continues.
4	Discard the message segment.
8	Terminate processing for the transaction.
12	Destination is invalid. Status AX will be returned to the application program and a 67D0 log record will be issued indicating error return code X'24'.

IMS treats any other return code as if it were 0, and processing continues.

Related reference

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

OTMA User Data Formatting exit routine (DFSYDRU0)

The OTMA User Data Formatting exit routine can determine and change the final destination of OTMA messages. The DFSYDRU0 exit routine can also format the User Prefix section of an OTMA asynchronous output message.

The DFSYDRU0 exit routine can change the final destination of OTMA messages by specifying OTMA member names, transaction pipe (Tpipe) names, or names of remote IMS systems.

You can specify OTMA C/I to use the HOLDQ when asynchronous output is created before the OTMA C/I client has established via client-bid. This is optional, as any queued output is moved by OTMA to the HOLDQ once the OTMA C/I client has connected and specified it is HOLDQ capable.

You can use the OTMA destination descriptor to avoid coding this user exit. See DFSYDTx in *IMS Version 15.2 System Definition* for full details on specifying OTMA descriptors.

Subsections:

About this routine

The following rules apply for this exit routine:

- This routine is optional.
- This routine is not called if the destination is an IMS scheduler message block (SMB) name.
- This routine cannot override the originating LTERM name.
- This routine can only set the final destination once.

  If output is routed from one OTMA client to another, that client cannot use its own Destination Resolution exit routine to set a different final destination.

Recommendation: Within a shared-queues group, ensure that the DFSYDRU0 exit routine is the same for both front-end and back-end IMS systems. If these exit routines differ on 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.

An OTMA client should not use a transaction name as a transaction pipe name (or routing key) because of potential conflict with the SMB name.

In a single IMS, multiple OTMA Destination Resolution exit routines are allowed. To display the DFSYDRU0 exit routine associated with an OTMA client, issue the /DISPLAY TMEMBER command.

IMS identifies the OTMA User Data Formatting exit routine for an OTMA client by searching, in the order listed, the following:

1. The exit routine specified on the client-bid call
2. The OTMA client descriptor
3. The default exit routine name, DFSYDRU0, if it exists

The exit routine specified on the client-bid call overrides the OTMA descriptor. The OTMA descriptor overrides the default exit routine name. If the default exit routine name does not exist, the OTMA User Data Formatting exit routine is not used.

If the message destination is the IMS Connect client that initiated the commit-then-send send-receive call and the SENDALTP function is activated, the message is sent back to the originating IMS Connect client through the incoming TPIPE.

The following table shows the attributes of the OTMA User Data Formatting exit routine.

Table 92. OTMA User Data Formatting exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	Different clients can have different exit routine names, or the clients can all use the default exit routine name of DFSYDRU0.
Binding	The OTMA User Data Formatting exit routine must be reentrant. The OTMA User Data Formatting exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. This exit routine is optional.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is eligible to use IMS callable services.
Sample routine location	IMS.ADFSSMPL (member name DFSYDRU0).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers at entry

At entry, the exit routine must save all registers using the provided save area. The registers contain the following information:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5
R13	Save area address (points to a single SAVEAREA, not a SAVEAREA chain)
R14	Return address
R15	Entry point address

This exit routine uses the Version 6 standard exit parameter list. The address of the work area that is passed to this exit routine in SXPLAWRK can be different each time that this exit routine is called.

The following table describes the contents of the OTMA User Data Formatting exit routine parameter list. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 93. Contents of the OTMA User Data Formatting user exit parameter list

Offset (decimal)	Contents
+0	Name of the originating LTERM or OTMA transaction pipe.

Table 93. Contents of the OTMA User Data Formatting user exit parameter list (continued)

Offset (decimal)	Contents
+8	The 8-byte destination name. If the destination is for OTMA and no tpipe name is specified in the output area, this field is used as the name of the tpipe to queue and deliver the output message. When a matching destination descriptor with EXIT=YES is found and the DFSYPRX0 exit routine either sets RC=100 or does not exist, this field contains the tpipe name that is specified in the destination descriptor.
+16	Transaction name or program name.
+24	Flag byte: **X'80'** An OTMA prefix exists. **X'40'** The user exit can override the client name. **X'20'** OTMA message submitted by OTMA client with super member support. The OTMA state data pointed to by the input parameter list has the 1-4 bytes super member name at offset x'E' from the beginning of the state data. **X'10'** The user exit is called to process a late response to a synchronous program switch request. If the X'80' flag is also set, the LTERM or tpipe name and input client member name in the parameter list are propagated from the original OTMA transaction that initiated the ICAL call. **X'08'** The destination name matches an entry in the OTMA destination descriptor for IMS Connect. **X'04'** The destination name matches an entry in the OTMA destination descriptor for IBM MQ. **X'02'** The destination name matches an entry in the OTMA destination descriptor for a non-OTMA destination.
+25	Synchronization level.

Table 93. Contents of the OTMA User Data Formatting user exit parameter list (continued)

Offset (decimal)	Contents
+26	Destination type flag: **X'80'** Transaction pipe exists for the client. **X'40'** LTERM exists in IMS (non-maintenence versions). **X'20'** LU 6.2 descriptor exists. **X'10'** ETO is available. **X'08'** Client is active. **X'04'** tpipe trace is active. **X'02'** This DRU exit was entered for a different OTMA client destination which is specified by a previous DRU exit.
+27	A 1-byte field that indicates the version of the exit routine parameter list: **X'80'** If set, indicates that at offset +100, the parameter list includes the address of the information from the OTMA destination descriptor for IBM MQ or IMS Connect. **X'40'** If set, indicates that at offset +104, the parameter list includes the original CHNG call value. **X'20'** If set, indicates that at offset +116, the user exit parameter list includes the address of the OTMA security data prefix. **X'10'** If set, indicates that the SENDALTP function is on for IMS Connect input message that triggers this ALTPCB output.
+28	User ID.
+36	Group name.
+44	The 16-byte name of the destination OTMA client. When a matching descriptor with EXIT=YES is found and the DFSYPRX0 exit routine either sets RC=100 or does not exist, this field contains the tmember name that is specified in the destination descriptor.
+60	Address of the PST block.
+64	Name of the originating OTMA client, if the message originated from an OTMA client; otherwise zeros.

Table 93. Contents of the OTMA User Data Formatting user exit parameter list (continued)

Offset (decimal)	Contents
+80	Address of the input Message Control Information prefix section of the OTMA message. If the exit is called to process a synchronous program switch response and the original transaction is from OTMA, this message control information is generated by IMS. The original message prefix from the OTMA client is not propagated to the user exit. However, the LTERM or tpipe name and the input client member name in this parameter list are from the original OTMA message.
+84	Address of the input State Data prefix section of the OTMA message. Check the prefix flag in the Message Control Information section to determine the specific type of State Data section specified. If the OTMA super member feature is used, the super member name is located at offset +14 from the beginning of the state data. See the TMAMSPNM field of the DFSYMSG macro. If the exit is called to process a synchronous program switch response and the original transaction is from OTMA, this state data information is generated by IMS. The state data prefix from the OTMA client is not propagated to the user exit. However, the correlator field (TMAMHCOR), the LTERM or tpipe name, and the input client member name in this parameter list are from the original OTMA message.
+88	Address of the input User Data prefix section of the OTMA message. The area is also used to return new or modified user data, up to a maximum of 1024 bytes.
+92	Address of the SCD block.
+96	Address of the output parameter list. This parameter list is used to return information to IMS. The contents of the output parameter list are shown in the following table.
+100	Address of the routing information defined in the OTMA destination descriptor for IBM MQ and IMS Connect destinations. If the destination name matches a non-OTMA destination descriptor, or the name does not match any entry in the OTMA destination descriptor, this field contains 0. See the TMAMICON_DESCRIPTOR DSECT mapping for the layout of the routing information for an IMS Connect destination. See the TMAMMQS_DESCRIPTOR DSECT mapping for the layout of the routing information for an IBM MQ destination.
+104	The 8-byte destination name from the original CHNG call. If the name is less than 8-bytes, it is left-justified and padded with blanks. This is an entry parameter only.
+116	Address of the OTMA security data, which is available in the input message prefix. This is an entry parameter only.

Contents of the output parameter list

The following table shows the contents of the output parameters list.

Table 94. Contents of the output parameter list

Offset (decimal)	Contents
+0	The 16-byte client override name, if any. This field is used when the destination is a different OTMA client. A return code of 8 must also be set.
+16	Output flag. **X'80'** If this flag is set, a synchronized transaction pipe must be created. However, if the OTMASP initialization parameter is set to Y in the DFSPBxxx PROCLIB member, the synchronized transaction pipe is always created. This flag can be set only if the return code is 0. **X'40'** If this flag is set, the message is persistent and a recoverable sequence number must be set. This flag is valid only if a synchronized transaction pipe is specified. **X'20'** If this flag is set, OTMA ensures that the output is always saved in a hold queue. This is an optional flag that is used only in the following scenario: In the shared-queues back-end IMS, the first call to DRU exit is to process a message from a non-OTMA client instead of an OTMA hold queue capable client. However, without this flag set in this scenario, the output is stored in the regular tpipe queue instead of in the hold queue **X'10'** If this flag is set, the ALTPCB output message will be sent to a remote IMS system for processing through an IMS Connect to IMS Connect TCP/IP connection. You must build a user data prefix that includes at least the names of both the destination remote IMS system and the remote IMS Connect that supports the remote IMS system. **X'08'** If this flag is set, the IMS system needs to send this ALTPCB output message back to the input IMS Connect client that initiates the commit-then-send send-receive call through the input tpipe.

Table 94. Contents of the output parameter list (continued)

Offset (decimal)	Contents
+17	A 1-character value with option F, C, or N. This value sets the security flag of the message prefix in an OTMA ALTPCB output message that is sent to a remote IMS system through IMS-to-IMS TCP/IP communications. The remote IMS system uses the security flag setting in the output message only if the input message does not have a security flag, there is no match OTMA destination descriptor found, and the remote IMS OTMA security is set to PR0FILE. **F** OTMA RACF security is set to FULL. **C** OTMA RACF security is set to CHECK. **N** OTMA RACF security is set to NONE.
+18	Reserved (2 bytes).
+20	The 8-byte tpipe name, if any. This field specifies the name of the tpipe that is used for queuing and transmitting the output message. If it contains all blanks, the destination name is used for the tpipe name. (This field is valid only when the return code is 0.) If the SENDALTP function is activated for this ALTPCB output message, this tpipe name is omitted.
+28	The 4-byte super member name. This field specifies the name of the super member and is used only in the following scenario: In the shared queues back-end IMS, the first call to DRU exit is to process a message from a non-OTMA client instead of an OTMA hold queue capable client. And, the output is destined to a hold queue capable client which supports super member. This field is valid only when the return code of DRU exit is set to 0 and the output flag byte is set to X'20'.

Contents of registers at exit

Before returning to IMS, the exit routine must restore all registers, except register 15, which must contain one of the following return codes:

Return code	Meaning
X'00'	Destination is the OTMA TPIPE. When the destination client name is from a matching destination descriptor with EXIT=YES, the descriptor defines the default destination for output message. However, if any value in the output parameter list is set, the user exit then determines the output destination instead of the descriptor.
X'04'	Destination is a non-OTMA LTERM.
X'08'	Destination is a different OTMA client (need to specify). The new client DRU0 exit will be invoked. **Note:** The OTMA Destination Resolution user exit (OTMAYPRX) can make all routing decisions by setting OTMAMD=Y in the DFSPBXXX PROCLIB member.
X'0C'	Destination is invalid. AL status on CHNG call. Can also be used to indicate any error in module processing.
X'64'	The matching OTMA destination descriptor determines the destination and the user data prefix in the output message. This return code is accepted when **EXIT=**YES is specified in the matching OTMA destination descriptor and the previous DRU exit, if used, does not specify a different OTMA client.
X'65'	The matching OTMA destination descriptor only determines the destination. The DFSYDRU0 exit routine determines the user data prefix in the output message. Any value in the output parameter list is ignored. This return code is accepted when **EXIT=**YES is specified in the matching OTMA destination descriptor and the previous DRU exit, if used, does not specify a different OTMA client.

Any other return code causes IMS to generate a DFS2370I message.

Error conditions

An A1 status code will be returned to the application program when the following errors occur:

- Incorrect 16-byte OTMA client override name is specified. The client name cannot contain all blanks or zeroes. If the client name is shorter than 16 bytes, it must be padded with blanks.
- The length of modified OTMA user data is over 1K.
- Incorrect return code is specified for the exit.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“IMS standard user exit parameter list” on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

OTMARTUX: OTMA Resume TPIPE Security user exit (DFSYRTUX and other OTMARTUX type exits)

The OTMA Resume TPIPE Security user exit (OTMARTUX) provides one of two possible layers of security for RESUME TPIPE calls that are issued to retrieve messages queued to the OTMA asynchronous hold queue.

This topic contains Product-sensitive Programming Interface information.

This level of security authorization interfaces with SAF and RACF only if the default resource class, RIMS, is defined to RACF. IMS installations can use this exit routine to authorize both the user ID and the transaction pipe name that is in the Resume TPIPE call message, to receive the output contained in the Resume TPIPE call message, in order to receive the output messages before any of these messages are sent to an OTMA client.

If the OTMARTUX user exit is defined to IMS, it is invoked, regardless of whether the first level authorization procedure is performed.

Subsections:

About this routine

The OTMA Resume TPIPE Security exit is invoked when a RESUME TPIPE call is received by OTMA if the user exit exists in the appropriate library. There are two security procedures with regard to TPIPE name and user ID authorization:

- RACF security procedure

  Verifies the existence of RACF resource name, RIMS or R*xxxxxxx*, where *xxxxxxx* is the value from the RCLASS EXEC parameter, the DFSPBxxx PROCLIB member or the DFSDCxxx PROCLIB member, and RACF authorization of the Resume TPIPE name and user ID combination.
- User exit security procedure
  - Invokes the OTMARTUX user exit. Your exit might take the result of the RACF security procedure, override its result, or add more restrictive security rules.

If authorization is successful, output messages in the hold queue are returned to the OTMA client. A rejection message (NAK) of the RESUME TPIPE call is sent to the client if authorization fails. If the user exit is not modified (that is, invoked as a passthru), the value of the sense code and the reason code in the message prefix remains the same. The return code in register R15 and the reason code in register R0 are the values from the first security procedure if performed. If not, register R15 and register R0 should contain zeroes. If the user exit is modified, you can complement the RACF security procedure or ignore it.

Attributes of the routine

The exit routine can serve the following functions for OTMA input and output messages:

- Override the results of the SAF and RACF interaction.
- Function as stand-alone resume transaction pipe security.
- Complement or supplement the security that is defined to RACF.
- Be invoked as a passthru module.

The following table shows the attributes of the OTMA Resume TPIPE Security exit routine.

Table 95. OTMA Resume TPIPE Security exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.

Table 95. OTMA Resume TPIPE Security exit routine attributes (continued)

Attribute	Description
Naming convention	You can name this exit routine DFSYRTUX and link it into a library that is included in the STEPLIB concatenation. If DFSYRTUX is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSYRTUX is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.
Binding	The OTMA Resume TPIPE Security exit routine must be reentrant.
IMS callable services	This exit routine is eligible for callable services. To use IMS callable services with this exit routine, examine the value of the SXPLATOK field in the "IMS standard user exit parameter list" on page 5: • If SXPLATOK is zero, you cannot use IMS callable services with this exit routine. • If SXPLATOK is non-zero, the value is the callable services token for this exit routine. You can use the 256-byte work area addressed by the SXPLAWRK field to call DFSCSIF0.
Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSYRTUX. If you use want the exit to be refreshable, specify EXITDEF=(TYPE=OTMARTUX,EXIT=(exit_names)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
Sample routine location	IMS.SDFSSMPL (member name DFSYRTUX).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers at entry

Normal linkage convention is used on entry and exit to and from this routine.

Register	Contents
R0	Reason code
R1	Address of the "IMS standard user exit parameter list" on page 5
R13	Save area address
R14	Return address
R15	Entry point address

Standard exit parameter list

This exit routine uses the Version 6 standard exit parameter list. The address of the work area that is passed to this exit routine in SXPLAWRK can be different each time that this exit routine is called.

If your OTMARTUX user exit can be called in an enhanced user exit environment, additional user exit routines might be called after your routine. When your user exit routine finds a transaction upon which to

act, it can set SXPL_CALLNXTN in the byte that SXPLCNXT points to. This tells IMS to not call additional exit routines.

Contents of registers at exit

Return code	Meaning
0	Authorization is successful. TPIPE protected.
4	Authorization is successful. TPIPE unprotected. all users/groups allowed access
8	Authorization Failure. See list of reason codes below under RTUPRSNC.

If register R15 is X'04', the sense code in the message prefix TMAMCSNC is X'33'. This sense code indicates that there must be a reason code in the message prefix TMAMCRSC. The applicable reason codes are listed in the following table under RTUPRSNC.

The following table describes the parameter list (DFSYRTUP) for the OTMA Resume TPIPE Security exit routine.

Table 96. Contents of the interface, DFSYRTUP

Label	Description
RTUPVERS	Version number of the parameter list.
RTUPTPNM	Address of the TPIPE name.
RTUPUSID	Address of the user ID. If this address is zero, there is no user ID (user ID is provided by the client).
RTUPSENC	Address of the sense code. The sense code for a failure in Resume TPIPE authorization is X'33'.
RTUPRSNC	Address of the reason code. The following reason codes are possible: • X'01': Security header was not provided in the message prefix • X'02': User ID was not provided in the message prefix • X'03': Group ID was not provided in the message prefix • X'04': User token was not provided in the message prefix • X'05': TPIPE name was not provided in the message prefix • X'06': RACF system failure • X'07': RACF security violation; no profile was defined for the user • X'08': User ID or Group ID is not authorized
RTUPRRET	Address of the return code from RACF. If this address is zero, the SAF parameter area does not exist.
RTUPRREA	Address of the reason code from RACF. If this address is zero, the SAF parameter area does not exist.
RTUPSFRC	Address of the return code from SAF. If this address is zero, the SAF parameter area does not exist.
RTUPSFRS	Address of the reason code from SAF. If this address is zero, the SAF parameter area does not exist.
RTUPSAFP	Address of SAF.
RTUPAMCI	Address of MCI.
RTUPASTD	Address of the prefix for state data.

Table 96. Contents of the interface, DFSYRTUP (continued)

Label	Description
RTUPASEC	Address of the prefix for security data. If this address is zero, there is no security data section in the prefix provided by the client.
RTUPINRC	Return code at entry to the exit routine.
RTUPINRS	Reason code at entry to the exit routine.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

PGMCREAT user exit routine type

The Program Creation user exit routine can request that IMS dynamically create the runtime program resource for a program that is to be scheduled in a BMP or JBP dependent region.

Optionally, the exit can also request that IMS create a runtime database resource. The program and optional database resource are created without requiring a restart of IMS or a type-2 CREATE command to be issued.

Subsections:

About this routine

During the scheduling process for a BMP or JBP, IMS checks to see whether the program resource has been defined to IMS. If the program resource has not been defined to IMS and a Program Creation user exit is defined, IMS calls the Program Creation user exit (PGMCREAT) to determine if the program resource should be created. If the PGMCREAT user exit indicates to IMS that the program resource should be created, IMS creates the program resource, along with the database resource if it was also requested, and then continues with the scheduling process. If the PGMCREAT user exit does not indicate to IMS that the program resource should be created or if the create of the program resource fails, the dependent region abnormally terminates.

The PGMCREAT user exit only supports the creation of a non-Fast Path BMP type program (FP(N), (BMPTYPE(Y)).

Dynamic resource definition (DRD) is required for this environment (MODBLKS=DYN). Only one optional database can be created.

Table 97. PGMCREAT exit routine attributes

Attribute	Description
IMS environment	DB/DC, DBCTL, DCCTL
Naming convention	User can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.

Table 97. PGMCREAT exit routine attributes (continued)

Attribute	Description
Binding	This exit routine must be reentrant. It executes in non-cross-memory mode under CTL TCB.
Including the routine	If you write your own exit routine and plan to use IMS callable services, you must manually link edit the routine with DFSCSI00. The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
IMS callable services	To use callable services with this routine examine the value of the SXPLATOK field in the IMS standard user exit parameter list to see if a callable services token is passed to the routine. If SXPLATOK is zero, you cannot use callable services with this routine. If SXPLATOK is non-zero, the callable services token is included, and you can use callable services. You can use the 256-byte work area addressed by SXPLAWRK in the standard user exit parameter list to call DFSCSIF0.
Sample routine location	IMS.ADFSSMPL (member name DFSPGCX0). The mapping of the DFSPGMCR interface block is available from the IMS library IMS.ADFSMAC (member name DFSPGMCR).

Processing options

The following processing options are valid for the PGMCREAT user exit. If you request an option that is not valid, IMS ignores your request and continues normal processing (the default option).

Creation of programs

To indicate to IMS that a program should be created, the exit must set the PGMCR_FC_CREATEPGM bit in the PGMCREAT parameter list which is mapped by the DFSPGMCR DSECT. The exit can specify the attributes of the program by setting the appropriate bits in the PGMCREAT parameter list. Any attribute values not specified will be obtained from the default program descriptor.

The PSB that is associated with the program must reside in ACBLIB or the IMS CATALOG.

Creation of databases (optional)

Database creation is optional. If a database resource is to be created, the name of the database must be set in the PGMCR_DBN field of the PGMCREAT parameter list. The exit can specify the attributes of the database by setting the appropriate bits in the PGMCREAT parameter list. Any attribute values not specified will be obtained from the default database descriptor.

The DBD associated with the database must reside in ACBLIB or the IMS Catalog.

Exporting resource definitions to the IMSRSC repository and RDDS

The program and database resources that are created by the PGMCREAT user exit can be defined to be exported by setting the PGMCR_FC_EXPORT bit in the PGMCREAT user exit parameter list.

If the exit indicates the resource definitions are not to be exported the resource definitions are not exported to the IMSRSC repository or system RDDS and are not available at next coldstart. In this case, after the next coldstart, if the program is scheduled in a BMP or JBP the PGMCREAT exit is called.

If the exit indicates the resource definitions are to be exported and IMS is defined to use the repository, the resources created by the PGMCREAT user exit are exported to the repository when one of the following conditions is satisfied:

- The names of the resources are specified with the NAME keyword or match the NAME keyword parameter that is specified on the **EXPORT DEFN TARGET(REPO)** command.
- An **EXPORT DEFN TARGET(REPO) OPTION(CHANGESONLY)** command is issued after the PGMCREAT user exit creates the resources.
- During the next autoexport if autoexport to IMSRSC repository is enabled. If AUTOEXPORT_IMSID=ALL then the resource definitions are exported to the IMSRSC repository for all IMS systems using the repository. In this case the program and or database may not exist locally at the other IMS systems. If AUTOEXPORT_IMSID=THIS_IMS then the resource definitions are exported to the IMSRSC repository for this IMS only.

If the exit indicates the resource definitions are to be exported and IMS is defined to use the system RDDS data sets, the resources created by the PGMCREATE user exit are exported to the system RDDS when one of the following conditions is satisfied:

- The names of the resources are specified or match with the NAME keyword on the **EXPORT DEFN TARGET(RDDS)** command.
- During the next system checkpoint if autoexport to RDDS is enabled.

Restrictions

If a program resource is created, the PSB associated with the program must reside in ACBLIB or the IMS Catalog. If a database resource is created, the DBD associated with the database must also reside in ACBLIB or the IMS Catalog.

Communicating with IMS

IMS uses the entry and exit registers, as well as a parameter list, to communicate with the exit routine.

Contents of registers on entry

Upon entry the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
Register	Contents
1	Address of “IMS standard user exit parameter list” on page 5.
13	Save area address.
14	Return address to IMS.
15	Entry point of exit routine.

All other registers are undefined.

“IMS standard user exit parameter list” on page 5

PGMCREAT exit routine parameter list

The address of the PGMCREAT parameter list is in the standard exit parameter list field SXPLFSPL. The PGMCREAT parameter list is mapped by DSECT DFSPGMCR.

Table 98. PGMCREAT exit parameter list

Field	Offset	Length	Field usage	Description
PGMCR_EYEC	X’00’	X’08’	Input	Eyecatcher Field

Table 98. PGMCREAT exit parameter list (continued)

Field	Offset	Length	Field usage	Description
PGMCR_VER	X’08’	X’04’	Input	Parameter version Number Value: **1** Version 1 parameter list. **PGMCR_VER1** Current PGM version.
	X’0C’	X’02’		Reserved
PGMCR_ECB	X’0E’	X’04’	Input	ADDR(ECB)
PGMCR_JOBN	X’12’	X’08’	Input	Job Name Field
PGMCR_PGMD	X’1A’	X’08’	Output	Program or descriptor name to use as a model to create a new program.
PGMCR_PGM	X’22’	X’08’	Input	Program name.
PGMCR_PSB	X’2A’	X’08’	Input	Program PSB Name.
PGMCR_PFLG1 • PGMCR_PF1_ DOPTY • PGMCR_PF1_ DOPTN • * • * • PGMCR_PF1_ GPSBY • PGMCR_PF1_ GPSBN	X’32’	X’01’	Output	Program Flag 1 • `X'80' - DOPT(Y)` • `X'40' - DOPT(N)` • `X'20' - Reserved` • `X'10' - Reserved` • `X'08' - GPSB(Y)` • `X'04' - GPSB(N)` • `X'02' - RESERVED` • `X'01' - RESERVED`

Table 98. PGMCREAT exit parameter list (continued)

Field	Offset	Length	Field usage	Description
PGMCR_PFLG2 • PGMCR_PF2_RESIDENTY • PGMCR_PF2_RESIDENTN • PGMCR_PF2_SCHDSER • PGMCR_PF2_SCHDPAR • PGMCR_PF2_TRANSTATY • PGMCR_PF2_TRANSTATN	X’33’	X’01’	Output	Program Flag 2 • X'80' - RESIDENT(Y) • X'40' - RESIDENT(N) • X'20' - SCHDTYPE(SERIAL) • X'10' - SCHDTYPE (PARALLEL) • X'08' - TRANSTAT(Y) • X'04' - TRANSTAT(N) • X'02' - Reserved • X'01' - Reserved
PGMCR_PFLG3	X’34’	X’01’	Input	Program Flag 3
PGMCR_PFLG4	X’35’	X’01’	Input	Program Flag 4

Table 98. PGMCREAT exit parameter list (continued)

Field	Offset	Length	Field usage	Description
PGMCR_PLANG • PGMC_ ASSEM • PGMCR_ COBOL • PGMCR_ JAVA • PGMCR_ PASCAL • PGMCR_ PLI	X’36’	X’01’	Output	Program language for Program Value: **01** Program Language set to Assembly Language. **02** Program Language set to COBOL. **03** Program Language set to JAVA. **04** Program Language set to PASCAL. **05** Program Language set to PLI.
PGMCR_PTYPE • PGMCR_ BMP • PGMCR_ JBP	X’37’	X’01’	Input	Program Type Flag • `X'80' - BMP program type` • `X'40' - JBP program type` • `X'20' - Reserved` • `X'10' - Reserved` • `X'08' - Reserved` • `X'04' - Reserved` • `X'02' - Reserved` • `X'01' - Reserved`
	X’38’	X’02’	Input	Reserved

Table 98. PGMCREAT exit parameter list (continued)

Field	Offset	Length	Field usage	Description
PGMCR_DBND	X’3A’	X’08’	Output	Database Descriptor name
PGMCR_DBN	X’42’	X’08’	Output	Database name
PGMCR_DFLG1 • PGMCR_DF1_RESIDENTY • PGMCR_DF1_RESIDENTN • PGMCR_DF1_ACCEXCL • PGMCR_DF1_ACCBRWS • PGMCR_DF1_ACCREAD • PGMCR_DF1_ACCUPD	X’4A’	X’01’	Output	Database Flag 1 • X'80' - RESIDENT(Y) • X'40' - RESIDENT(N) • X'20' - ACCTYPE(EXCL) • X'10' - ACCTYPE(BRWS) • X'08' - ACCTYPE(READ) • X'04' - ACCTYPE(UPDATE) • X'02' - Reserved • X'01' - Reserved
	X’4B’	X’03’	Input	Reserved
	X’4E’	X’12’	Input	Reserved

Table 98. PGMCREAT exit parameter list (continued)

Field	Offset	Length	Field usage	Description
PGMCR_FLGC • PGMCR_FC_EXPORT • PGMCR_FC_PGMDESC • PGMCR_FC_CREATEPGM • PGMCR_FC_DBDESC	X’60’	X’01’	Output	Program Control Flag • X'80' - EXPORT Indicator • X'40' - The name in PGMCR_PGMD program descriptor: – If 0, PGMCR_PGMD specifies a program name. – If 1, PGMCR_PGMD specifies a program descriptor name. • X'20' - Create program specified in PGMCR_PSB. • X'10' - The name in PGMCR_DBND Database descriptor: – If 0, PGMCR_DBND specifies a database name. – If 1, PGMCR_DBND specifies a database descriptor name. • X'08' - Reserved • X'04' - Reserved • X'02' - Reserved • X'01' - Reserved

Table 98. PGMCREAT exit parameter list (continued)

Field	Offset	Length	Field usage	Description
	X'61'	X'03'	Input	Reserved
	X'64'	X'12'	Input	Reserved

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers. Information is passed back to IMS using the PGMCREAT parameter list.

Physical Terminal (Input) edit routine (DFSPIXT0)

The Physical Terminal (Input) edit routine (DFSPIXT0) user-written edit routine gains control before the IMS Basic Edit routine. It is used to accept, modify, and cancel segments and messages.

This topic contains Product-sensitive Programming Interface information.

This chapter describes the Physical Terminal (Input) Edit routine. This user-written edit routine gains control before the IMS Basic Edit routine. If the input message is processed by MFS, the Physical Terminal (Input) edit routine is not called. This edit routine is called only when inserted from a terminal; it is not called when the message is inserted by a program-to-program switch. This edit routine is not called for LU 6.2 terminal input.

Subsections:

- "About this routine" on page 260
- "Communicating with IMS" on page 262

About this routine

Message segments are passed one at a time to the Physical Terminal Input edit routine, and the edit routine can handle them in one of the following ways:

- Accept the segment and release it for further editing by the IMS basic edit routine.
- Modify the segment and release it for further editing by the IMS basic edit routine. Examples of segment modifications that can be made are changing the transaction code and reformatting the message text. Make any required modifications, because IMS has not yet performed destination or security checking.
- Cancel the segment.
- Cancel the message and request that the terminal operator be notified accordingly.
- Cancel the message and request that a specific message from the User Message Table be sent to the terminal operator.

The Physical Terminal Input edit routine requests these actions by specifying different return codes that are interpreted and acted on by IMS.

The following table shows the attributes of Physical Terminal (Input) edit routine.

Table 99. Physical terminal (input) edit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must provide 1-byte to 8-byte name.

Table 99. Physical terminal (input) edit routine attributes (continued)

Attribute	Description
Binding	Prior to IMS 15, DFSPIXT0 was linked into the IMS nucleus, which automatically made it non-reentrant (because the nucleus is non-reentrant). In IMS 15, DFSPIXT0 was removed from the IMS nucleus and is now loaded as a stand-alone module during IMS initialization. Thus, if you use IMS 15 and you link this user exit as reentrant, you must ensure that it is not dependent on any information from a previous iteration and that it does not store into itself. To learn more, see Migration considerations for removing user exit routine specification from system definition (Release Planning) and Routine binding restrictions (Exit Routines).
Including the routine	The edit routine must be loaded into an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. Manually link this exit with DFSCSI00 to use callable services. No additional linking is required to use IMS callable services.
Sample routine location	IMS.ADFSSMPL. **Note:** The sample exit routine is not reentrant. You must assemble it with PARM='OBJECT,NODECK,NORENT' and link-edit it with PARM='NCAL,LET,LIST,XREF,SIZE(880K,64k)'.

Bypassing Basic Edit

If the IMS application program supplies DFS.EDTN in the MOD name parameter for the output message, the IMS basic edit routine will be bypassed except for transaction code and password validation.

Related Reading: For further information, see "MFS Bypass for the 3270 or SLU 2" in the "Application Programming Using MFS" chapter in *IMS Version 15.2 Application Programming APIs*.

The Physical Terminal Input edit routine must position the transaction code, and optionally the password, if the terminal is not operating in conversational or preset destination mode. The exit routine should detect errors and return a message to the terminal operator if any errors are found.

IMS maintains a flag in the CTB (bit CTB6TRNI in the CTBFLAG6 field) to indicate when 3270 MFS bypass, nonconversational, no preset destination, and first segment exist on input to the Physical Terminal Input exit routine. This flag notifies the Physical Terminal Input exit routine that it can add a minimum of 1 and a maximum of 18 bytes to the front of the message segment for a transaction code and optional password. The minimum of 1 byte to be added to the front of the message segment consists of a 1-byte transaction code. If NOBLANK is not specified at system definition, a minimum of 2 bytes is added to the front of the message segment, consisting of a 1-byte transaction code and 1 blank, which is necessary as a separator. To add a transaction code and optional password, the exit routine can put a return code of 16 in register 15 and set register 1 to point to an LLZZ field, followed by the data to be added.

Specifying the routine

The Physical Terminal Input exit routine (DFSPIXT0) is specified on the LINEGRP or TYPE macros as part of the EDIT parameter. If you are using **both** the Physical Terminal Input and Output edit routines, you must specify (YES,YES) on the EDIT parameter of the TERMINAL macro or Extended Terminal Option (ETO) logon descriptor.

The CSECT name for this edit routine is the name specified in the TYPE or LINEGRP macro statement for which this edit routine applies. You must also specify YES in the EDIT parameter of the TERMINAL macro statement or ETO logon descriptor.

The Global Physical Terminal Input edit routine (DFSGPIX0) performs the same functions as this edit routine but does not require system definition.

Related Reading:

For information on coding the LINEGRP, TYPE, and TERMINAL macros, see *IMS Version 15.2 System Definition*.

For more information on the ETO feature, see *IMS Version 15.2 Communications and Connections*.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of the input message segment buffer. IMS editing has not been performed. The first two bytes of the buffer contain the segment length (binary length includes the 4-byte overhead). The third and fourth bytes of the buffer are binary zeros. The message text begins in the fifth byte of the buffer. If the device was defined with MFS support but this message is not being processed by MFS, the first segment of the message has backspace error correction performed before entry to this edit routine. If escape (**) was entered by the terminal operator, the first two data bytes have been changed to binary zeros.
7	Address of CTB for the physical terminal from which the message was entered.
9	Address of CLB for the physical terminal from which the message was entered.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of edit routine.

The edit routine you supply can edit the message segment in the buffer pointed to by register 1.

You can reduce the length of the message segment to any size by replacing the length in the buffer with the appropriate value. The length field must appear in the same place at exit as at entry, and bytes 3 and 4 must not be changed.

Contents of registers on exit

On return to IMS, all registers must be restored except for register 1, which contains a message number if register 15 contains a value of 12; otherwise it is ignored. Register 15 contains one of the following return codes:

Return code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled, and the message identified by register 1 is sent to the terminal.

Return code	Meaning
16	Insert the transaction code and optional password following the LLZZ pointed to by register 1. This return code is only valid for 3270 MFS bypass terminals. When the entering terminal is not a 3270 MFS bypass terminal, and the physical terminal input exit gives a return code of 16, IMS issues an error message, and the transaction code is not inserted in the message.

Any other return code causes the message to be canceled and the terminal operator to be notified.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

"Global Physical Terminal (Input) edit routine (DFSGPIX0)" on page 174
The Global Physical Terminal (Input) edit routine (DFSGPIX0) is called before the IMS Basic Edit routine and eliminates the overhead associated with defining the edit routine for each terminal through system definition.

Sample Physical Terminal (Input) edit routine (DFSPIXT0)

Use the Sample Physical Terminal (Input) edit routine (DFSPIXT0) to test input message segments.

This routine performs the following functions:

- Scans the input message segment for an expected format (TESTEXIT)
- Generates return codes (XX) based on the input request (TESTEXIT,XX)
- Verifies the user message number (YYY) if specified (TESTEXIT,XX,YYY)
- Replaces TESTEXIT with ERROR if return code or message number is invalid and passes the segment to IMS (return code 0)

Physical Terminal (Output) edit routine (DFSCTTO0)

The Physical Terminal (Output) edit routine enables you to edit output messages immediately before they are sent to a terminal.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

During system definition, you specify which physical terminals or set of VTAM nodes use the defined edit routine for output editing. You can use these edit routines to meet your special editing needs associated with different communication terminals.

An output message can be processed by 1) a Physical Terminal Output edit routine and the IMS Basic Edit routine or 2) a Physical Terminal Output edit routine and MFS (Message Format Service). Output editing is performed in this sequence. Therefore, the input to the edit routine is the output of the application program, and the output of the edit routine is the input to MFS or the IMS Basic Edit routine.

You can also specify that this edit routine cancel an output message so that it is not delivered to the terminal. Instead, the routine can optionally request that an error message be sent in place of the canceled message.

The following criteria apply to message cancellation:

- Output messages can be canceled if they are destined for VTAM terminals only.
- Conversational output and IMS in-core system messages cannot be canceled. Such cancellation requests from the exit are ignored, and the output message is sent.
- The request to cancel must be made for the first segment of a message. Requests for non-first segments of a message to be canceled are ignored, causing normal output processing to continue for the message.
- This routine is not activated for messages going across an MSC VTAM link, so these messages cannot be canceled.

The following table shows the attributes of the Physical Terminal (Output) edit routine.

Table 100. Physical terminal (output) edit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must provide a 1-byte to 8-byte name.
Binding	Prior to IMS 15, DFSCTTO0 was linked into the IMS nucleus, which automatically made it non-reentrant (because the nucleus is non-reentrant). In IMS 15, DFSCTTO0 was removed from the IMS nucleus and is now loaded as a stand-alone module during IMS initialization. Thus, if you use IMS 15 and you link this user exit as reentrant, you must ensure that it is not dependent on any information from a previous iteration and that it does not store into itself. To learn more, see Migration considerations for removing user exit routine specification from system definition (Release Planning) and Routine binding restrictions (Exit Routines).
Including the routine	The edit routine must be loaded into an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. Manually link this exit with DFSCSI00 to use callable services. No additional linking is required to use IMS callable services.
Sample routine location	IMS.ADFSSMPL.

Specifying the routine

The Physical Terminal Output edit routine (DFSCTTO0) is specified on the LINEGRP or TYPE macro as part of the EDIT= parameter. If you are using **both** the Physical Terminal Input and Output edit routines, you must specify (YES,YES) on the EDIT= parameter of the TERMINAL macro.

Related Reading: For information on coding the LINEGRP, TYPE, and TERMINAL macros, see the section on "Macros" in *IMS Version 15.2 System Definition*.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry, the edit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	The address of a buffer containing the output message segment to be edited. The first two bytes are a binary count of the message segment length. The second two bytes are control information provided by the application program that constructed the message. The text of the output message starts in byte 5. The count includes the first four bytes in length. This register contains zeros if flag CTBAEOM in field CTBACTL of the CTB is on, indicating end-of-message. Any exit that modifies the contents of the buffer passed in register 1 should test for an end-of-message condition.
2	The address of an 8-byte field that contains either binary zeros or the user ID associated with the output message. The contents of the user ID field are described in *IMS Version 15.2 Application Programming* in the section on "I/O PCB Masks" in "Defining Application Program Elements". The user ID in the output message can be compared to the user ID in the CTB (CTBUSID) to determine editing requirements. The user ID is only checked on the first segment of a multisegment message. DFSCTTO0 uses CTBAEOM and ENTSTAT to determine which segment is being processed.
3	Address of storage area. For details of the format of this storage area, see the prolog in the sample routine (IMS.ADFSSRC; member name is DFSCTTO0).
7	CTB address for the destination terminal. **CTBFLAGC field**: CTBCDSDT bit *on* means that session restart has occurred for this terminal. If the edit routine is called with the CTBCDSDT on, the edit can assume that this is the first application output message selected for output processing since the session has restarted (provided that the bit is turned **off** by the edit routine after the first message is processed). IMS turns this bit **on** every time SDT (Start Data Traffic) for VTAM occurs. The edit routine is responsible for resetting this flag after it receives the first message. **CTBFLAG4 field**: CTB4RESP bit **on** means that the terminal is in response mode. After a system restart, IMS resets CTB4RESP.
9	Address of CLB. This block starts with a DECB. The content of the DECAREA field in the DECB is equivalent to the content of register 1.
11	Address of SCD.
13	Address of save area. The edit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

The output message segment that your edit routine returns to IMS from must be pointed to by the contents of the DECAREA field of the DECB. The first four bytes must be in a format as received at input with the binary count updated to the edited message segment length inclusive of the four bytes of prefix.

Contents of registers on exit (if no cancel request)

Before returning to IMS, the edit routine must restore all registers. If you are editing the message in place, you can increase its length by a **maximum** of ten bytes.

When the last segment of a message has been edited, IMS returns control to the routine. The routine has no new message data to edit.

Whenever a Physical Terminal Output edit routine is called, the CTB is in register 7. A 1-byte field, CTBACTK, in the CTB contains a 1 in the second bit position if this entry to the routine is for end of message (EOM).

Contents of registers on exit (if cancel request)

On return, registers must be restored except for register 15, which must contain the following return code.

Return code	Meaning
0	Message canceled without error message DFS3489. The buffer length must be set to zero.
4	Message canceled. The buffer length must be set to zero.

All registers are not restored when a cancel request is made and the edit requests that IMS send an error message DFS3489 to the terminal for a non-response-mode message.

In order for IMS to cancel an output message before it is sent to the terminal, the Physical Terminal Output edit routine must make a request when the first segment of an output message is presented to it. The edit makes this request by setting the length of the first segment to zero in the buffer pointed to by DECAREA.

If the edit routine wants IMS to send error message DFS3489 in place of the canceled message, it places a return code of 4 in register 15 (in addition to zeroing the length field of the first segment).

If the terminal is in response mode, IMS always replaces the canceled message with error message DFS3489. Across a system restart, response mode is reset. Therefore, if an output message is canceled after the system restart, no error message is sent.

If the terminal is not in response mode, the edit routine is not required to have IMS send error message DFS3489. Nevertheless, it might be necessary to have IMS send the error message to prevent a hang condition for certain device types that are expecting a message.

Related Reading: For an explanation of error message DFS3489, see *IMS Version 15.2 Messages and Codes, Volume 1: DFS Messages*.

Related reference

“Links with your exit routine and DFSCSI00” on page 16
To use callable services, your exit routine must be linked with the callable service interface module, DFSCSI00. You need to manually link this module to your exit routine.

“Initialization of IMS callable services (DFSCSII0)” on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample Physical Terminal (Output) edit routine (DFSCTTO0)

The Sample Physical Terminal (Output) edit routine (DFSCTTO0) shows how to extend an output message and how to attach a prefix.

IMS callable services are used to get and release storage. This example applies to single-segment or multisegment messages, and to as many devices as the edit routine's table is assembled to handle. The default table size allows for five devices, but can be changed by modifying the label NUMENTS. If the table capacity is exceeded, an ABENDU55 results. If the prefix had not increased the message length by more than ten bytes, it could have been attached without the creation of an additional buffer area.

Queue Space Notification exit routine (DFSQSPC0/DFSQSSP0)

The Queue Space Notification exit routine (DFSQSPC0) is activated and a message is issued when a logical record is assigned to or released from a message queue data set.

This topic contains Product-sensitive Programming Interface information.

This routine causes a message to be issued when one of following occurs:

- The number of records currently in use exceeds upper threshold percentage value of the maximum number assignable before initiation of automatic shutdown.
- The number of records currently in use falls below the lower threshold percentage value of the same maximum.

IMS sets an upper threshold value of 75 percent, and a lower threshold value of 60 percent. You can modify these values by using the QTU and QTL parameters of the IMS procedure.

QTU has a range of 2 percent through 100 percent, and QTL has a range of 1 percent through 99 percent.

The exit routine can also be called optionally when a BMP's unit of work is completed.

In a shared-queues environment, the Queue Space Notification exit routine is DFSQSSP0. The following information is passed when this exit routine runs.

- The shared queue structure is in an overflow state.
- The destination queue in a shared-queues environment is in an overflow state.

Subsections:

- "About this routine" on page 267
- "Restrictions" on page 269
- "Communicating with IMS" on page 269

About this routine

By using the SHUTDWN parameter of the MSGQUEUE macro, you can reserve a number of records in each message queue data set. If the data set fills up with unprocessed messages, the system automatically shuts down with an internal CHECKPOINT DUMPQ.

If unprocessed messages overflow a message queue data set before the automatic shutdown completes, a U0758 abend occurs.

This exit routine provides a warning before the automatic shutdown is initiated, so you can reduce message queue buildup, possibly avoiding the automatic shutdown and, most importantly, the U0758 abend.

You can replace the IMS-supplied exit routine with your own to establish your own threshold algorithm or issue user messages, which can then be captured by the AOI exit routine to reduce queue usage.

As an option, for certain units of work, you can modify this exit to find the number of records currently in use by the calling task. You can also request information that can be used to terminate the unit of work. For each application, LU 6.2 conversation, or OTMA session, IMS maintains counts of short and long message queue records (DRRNs) assigned, and supplies them to DFSQSPC0/DFSQSSP0 if this option is used.

If you use this option, the expanded parameter list contains an output field that allows you to tell IMS that you want the unit of work stopped because one or both of the counts have exceeded specified limits. Different count limits can be established for different tasks.

For most program types, the record counts are reset when one of the following occurs:

- A message is retrieved (GU call) from the message queues.
- A sync point occurs.
- A rollback occurs.

- The application terminates normally.

Exception: For non-message-driven BMPs, and for multiple-mode transactions (MODE=MULT specified on the TRANSACT macro), the queue counts are not reset until normal termination. If the unit of work is a DC transaction or conversation the counts are not provided.

For LU 6.2 conversations, the record counts are reset for each new message.

For OTMA sessions, the record counts are reset:

- For each new incoming message.
- For each IMS conversational iteration.
- When MSC remote output is received at the original IMS. The original IMS will then deliver the output to the OTMA client.

The exit routine terminates a unit of work in the following ways:

- For an application program, an 'A7' status code is returned to the application. If the AIBTDLI call interface was used, the application also gets an AIB return code (X'104') and a reason code (X'190'). If the application tries to insert a message after the unit of work terminates or to a destination that is stopped, the application immediately receives an 'A7' status code and the call is not processed.
- For an LU 6.2 conversation, message DFS0777I is sent and the conversation is deallocated.
- For an OTMA session, the OTMA input is rejected with a NAK message.

After the unit of work terminates, message queue records in use are released.

The sample exit routine DFSQSPC0/DFSQSSP0 (IMS.ADFSSRC) describes how to enable this option. Using this option creates some additional overhead from building the expanded parameter list. The default for this option is NO.

The following table shows the attributes of the Queue Space Notification exit routine.

Table 101. Queue space notification exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSQSPC0 (or DFSQSSP0 for shared queues).

Binding	This routine must be reentrant. It can be called in cross-memory mode.
	
	DFSCSI00 (callable services module) must be included in this load module if you plan to use IMS callable services from this exit routine. An example of the bind control statements is:
	
	`INCLUDE LOAD(DFSQSPC0)     SPACE NOTIFY USER EXIT`
	`INCLUDE LOAD(DFSCSI00)     IMS callable services`
	`MODE    AMODE(31),RMODE(ANY)`
	`ENTRY   DFSQSPC0`
	`NAME    DFSQSPC0(R)`

Including the routine	DFSQSPC0 is a separately linked composite module in the IMS.SDFSRESL. If you write your own exit routine, it must be linked into IMS.SDFSRESL.
IMS callable services	To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. The IMS-provided version of DFSQSPC0/DFSQSSP0 includes an example that uses callable services to obtain working storage during an initialization call to DFSQSPC0/DFSQSSP0. Use the ECB passed in parameter list QSPCECB for IMS callable services.

Table 101. Queue space notification exit routine attributes (continued)

Attribute	Description
Sample routine location	IMS.ADFSSRC (member name DFSQSPC0 or DFSQSSP0). The DFSPARM macro is in IMS.ADFSMAC (member name DFSPARM).

Exit routine call types

The following call types are recognized by the Queue Space Notification exit routine. Some of the parameters that are passed to the exit routine vary with the call type.

Call type	Description
1	Initialization call
2	Application assigned DRRN (for example, a DL/I application)
3	LU 6.2 assigned DRRN
5	Free the DRRN
6	OTMA assigned DRRN
7	MSC DRRN ASSIGN CALL
8	BMP unit of work completed (optional)

Restrictions

The following restrictions apply to DFSQSPC0/DFSQSSP0:

- z/OS services are unavailable to programs that are running in cross-memory mode unless the service's documentation specifically states that it is available.
- Code running in cross-memory mode cannot issue any SVCs except ABEND.

Because this exit is called every time a message queue data set record is assigned or released, the logic you add to this exit can have a negative effect on system performance. IWAITs, time-consuming algorithms, and excessive use of IMS callable services should be avoided.

If you want to issue user messages instead of IMS system messages DFS2013 through DFS2018, you must provide an exit that returns user message keys in register 15. The value that is returned in register 15 is actually the negative of the key in the user message table. In addition to returning the appropriate message key in register 15, ensure that the message text is in the user-supplied message table, DFSCMTU0.

Communicating with IMS

The queue space notification exit routine is called whenever a logical record is assigned to or released from a message queue data set. A parameter list is passed to the exit routine. Its contents depend on whether the user-provided DFSQSPC0/DFSQSSP0 takes advantage of the optional capabilities that are provided by IMS. The IMS-provided DFSQSPC0/DFSQSSP0 does not use the optional capabilities, although it does describe how they can be used.

To take advantage of the optional capabilities, you must modify DFSQSPC0/DFSQSSP0 to recognize the initialization call type (Type 1). When the call is made, if bit QSPCF2IN in the parameter list field QSPCFLG2 is turned on, IMS will set a flag in the SCD. The SCD flag tells IMS to provide the expanded parameter list to DFSQSPC0/DFSQSSP0. To activate optional call type of BMP, set bit QSPCF3BT in the parameter list field QSPCFLG3. This will set a flag in the QSCD that tells IMS to call the exit when a BMP Unit of Work has completed. The INIT call is made only during early IMS (Queue Manager) initialization to enable the user exit to obtain working storage that is always to be available to DFSQSPC0 through the parameter list.

User-provided versions of DFSQSPC0/DFSQSSP0 need not change if the message record count capability is not used.

The parameter list is mapped by the macro DFSPARM. The parameter list has the following parts:

1. Message queue data set in-use count and threshold status
   - The number of records currently in use

     The high-order byte of the in-use count is used as a flag byte.
   - The maximum number of records assignable before shutdown (not provided for shared queues)

     The exit routine interrogates these values and sets the parameter flag and a return code (register 15) based on their values. The return code is either zero or an error message number.
2. Pointers to control blocks and thresholds

   These fields are always passed to DFSQSPC0/DFSQSSP0:
   - Address of the SCD control block
   - Address of the ECB (required for IMS callable services).
   - Address of user exit's work area or zero
     - During the initialization call to DFSQSPC0/DFSQSSP0, you can use IMS callable services to obtain working storage for your exit. The address that you store in the parameter list is saved by IMS and returned to your exit on every call. IMS only saves the address returned by the exit during the initialization call. Addresses that are returned during other calls are overlaid by the address that is returned from the initialization call, or by zero if no address was returned.
     - User-provided exit is responsible for obtaining the work area during the initialization call to DFSQSPC0/DFSQSSP0 and storing its address in the parameter list.
   - Upper and lower threshold limits (Same values as found in QSCDQTU and QSCDQTL). The threshold values will not be set on a call type of 8.
   - DFSQSSP0 is passed the same fields as DFSQSPC0, except for the upper and lower threshold limits.

   The parameter list fields QSPCQTU and QSPCQTL contain the upper and lower threshold values (DFSQSPC0 only). The thresholds are either:
   - IMS defaults (75 percent and 60 percent).
   - Your default values specified in QTU and QTL in the DFSPBxxx member.
   - QTU and QTL values specified in the IMS procedure.
3. Type of Call and other input/output flags

   The following fields are only used while processing call types: 1, 2, 3, 5, 6, and 8. In all other cases, the call type is set to zero. For call type 8, only the call type is set.
   - Call type
   - Assign/Free DRRN indicator
   - Message Queue record count exceeded flag - set by exit

   The following flags can be set if shared queues are active (DFSQSSP0):
   - Shared queue structure is in an overflow state.
   - Destination queue is in an overflow state.
4. Unit of work information

   These fields are only used while processing a DL/I (Call Type 2) application, an LU 6.2 (Call Type 3) terminal request, or an OTMA (Call Type 6) request:
   - Accumulated counts of short and long message queue records assigned by this unit of work
   - Identification of the unit of work making the call:

     For a DL/I application: TRAN name, PSB name, and Terminal Symbolic

For an LU 6.2 Terminal: LU name, TP name and length, Side name

For an OTMA client: Tpipe name, z/OS cross-system coupling facility member name, and override LTERM name

5. Message destination name

   The message destination name is used while processing Call types 2, 3, 5, and 6. If the destination name is not available at the time of the call, this field is set to zero.

6. Structure usage feedback data (DFSQSSP0 only)

   The feedback data is the usage information about the primary structure and overflow structure of IMS shared queues .

   The feedback area is optional for the exit. By default, IMS does not provide the feedback data. The exit must set the flag QSPCF3BT in QSPCFLG3 during the initialization call (type 1 call) to request that the feedback data be passed on subsequent exit calls.

   When requested, the feedback data is passed to the exit during call types: 2, 3, 5, 6, and 7. Meanwhile, IMS sets QSPCF3BT in QSPCFLG3 to indicate that the feedback area is being sent.

   QSPCFBKP points to the beginning of the structure feedback area. The feedback area contains the message queue structure usage information that is returned from the last CQSPUT or CQSDEL request issued by IMS. The format of the feedback area is described by mapping macro CQSSFBA (CQS Structure Feedback Area). For details about the feedback area, see the CQSSFBA macro shipped with IMS.

   Considerations when using the feedback area pointed to by QSPCFBKP:

   - The data in the feedback area is volatile. It is updated on each CQSPUT and CQSDEL call issued by IMS, and the values in it might change asynchronously while the Queue Space Notification exit is running. If your exit needs unchanging values, you must copy the fields that you need into your own storage or into registers to ensure that they do not change while you are using them.
   - The data in the feedback area can be outdated. For example, one IMS/CQS system might access queues that are on the overflow structure, while another IMS/CQS system might not access overflow queues. The second IMS/CQS system might not have the most current utilization data for the overflow structure because it has not accessed the overflow structure recently.
   - The data in the feedback area can be zero. DFSQSSP0 might be called by IMS after the initialization call (where it requests CQS structure usage statistics), but before IMS has made the first CQSPUT or CQSDEL call to populate the CQS structure feedback area. You can detect this situation by checking the first byte of the feedback area ID string SFBA_ID for binary zeros. The field will be zero if no CQSPUT or CQSDEL access has been made, and you should not use any of the CQS structure usage statistics fields. It will be nonzero after the first CQSPUT or CQSDEL call has been made.

   QSPCFBKL field contains the structure feedback area length in bytes. The data beyond the feedback size is unpredictable.

The IMS-supplied Queue Space Notification exit routine, which is found in IMS.ADFSSRC, can be used as a guide in creating your own exit routine.

If you want to change the threshold notification algorithm, note the following interface requirements.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents	
0	Data set indicator:	
	00	QBLKS data set or Call Type 8. For Call Type 8, there is no data set indicator value to put in Register 0.
	04	SMSGQ data set
	08	LMSGQ data set
2	Address of parameter list	
9	Address of ECB	
10	Address of SCD	
14	Return address to IMS	
15	Entry point of exit routine	

Description of parameters

The macro DFSPARM generates the DSECT for the parameter area passed to DFSQSPC0/DFSQSSP0 by IMS. For more information, refer to the DFSPARM macro included in IMS.ADFSMAC.

A pointer to the SCD is contained in the input field QSPCSCD as well as in Register 10.

Recommendation: Get addressability to the SCD from the parameter list rather than register 10.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15. Register 15 must contain one of the following return codes, except for call type 8, which does not check for a return code.

Return code	Meaning
0	No message is issued
Message key	Queue Manager issues a message
Negative	User-defined message is sent

Related concepts

“Guidelines for writing IMS exit routines” on page 3
Use the guidelines in this information to write IMS exit routines, enable IMS exit routines to perform functions with callable services, and reference all callable service return and reason codes.

Recovery-related EXEC parameters for the control region (System Definition)

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“Initialization of IMS callable services (DFSCSII0)” on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

“User Message table (DFSCMTU0)” on page 475

You can create your own messages and list them in the User Message table (DFSCMTU0).

Security Reverification exit routine (DFSCTSE0)

The Security Reverification exit routine (DFSCTSE0) allows you to reevaluate transaction authorization checking on the DL/I CHNG Call.

Transaction Manager applications that use Multiple System Coupling (MSC), CHNG calls, and AUTH calls on a remote IMS system can benefit from coding this exit routine. By coding this exit routine, you can avoid a security failure that occurs when RACF or a non-RACF security environment is called in a destination MSC system by a user that is not signed on to that particular IMS system.

This routine is optional when IMS dynamically creates a security environment in a remote IMS back-end system (or a local IMS system if the user has signed off) to accomplish the RACF authorization check. You can control the creation of the security environment by using the Build Security Environment user exit (BSEX). IMS calls this routine, if available, to provide compatibility.

The IMS security exit routines do not need to be bound to the IMS nucleus, can run in 31-bit storage, and can share a work storage area. The following security exit routines have these attributes:

- Signon/off security exit routine (DFSCSGN0)

  DFSCSGN0 is called during IMS initialization to give the exit routine the chance to acquire a work storage area. The exit routine passes the address back to IMS. Then, IMS passes the address to the other security exit routines every time they are called.
- Security Reverification exit routine (DFSCTSE0)
- Transaction Authorization exit routine (DFSCTRN0)

Subsections:

- "About this routine" on page 273
- "Communicating with IMS" on page 274

About this routine

This exit routine is an entry point in DFSCTRN0. If you do not code this entry point, IMS does not call it.

The following table shows the attributes of the Security Reverification exit routine.

Table 102. Security reverification exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCTSE0.

Table 102. Security reverification exit routine attributes (continued)

Attribute	Description
Binding	DFSCTSE0 can be bound to DFSCTRN0, or coded as an explicit part of DFSCTRN0. If you code this entry point, it must have access to a table of valid user IDs, passwords, and transactions associated with each valid user ID, or contain some algorithm to derive this authorization information. For addressability, this table must reside in this module, in the /SIGN ON exit (DFSCSGN0), or in the IMS nucleus. In IMS Version 12 and earlier, the security exit routines must be bound to the IMS nucleus because the SECURITY macro is included in the IMS nucleus. In IMS Version 13 and later, the SECURITY macro is not supported and the security exit routines can be bound separately. If the security exit routines are linked in one of the STEPLIB or LINKLIST libraries, IMS loads the exit routine. There is no startup parameter to specify whether to load the routines. Message DFS1937I is issued for every exit routine that is loaded into 31-bit storage. If the exit routines cannot be linked separately or cannot use a common work area, they must be linked in the following manner: • If the CSECT of DFSCTSE0 is part of DFSCTRN0 source, DFSCTSE0 must be linked as an ALIAS of DFSCTRN0. • If virtual address spaces are used to exchange data between DFSCSGN0, DFSCTRN0, and DFSCTSE0, you must link DFSCTSE0 and DFSCSGN0 as ALIASs of DFSCTRN0.
Including the routine	If DFSCTSE0 is link edited to DFSCTRN0, it is called on return from DFSCTRN0.
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found in register 9 for the DFSCSII0 call. • Link DFSCSI00 with your user exit routine.
Sample routine location	No sample is provided.

Communicating with IMS

You can call the exit routine in the following environments:

- If you are operating in a RACF environment and the RACF FRACHECK call returns a valid return code (0 or 4), IMS immediately calls the exit routine DFSCTRN0 if DFSCTRN0 exists. On returning from DFSCTRN0, regardless of its return code, IMS immediately calls DFSCTSE0 if DFSCTSE0 exists. This applies to AUTH and CHNG calls only.
- If you are operating in a non-RACF environment and DFSCTSE0 is coded as an entry point, IMS calls this entry point following each call to DFSCTRN0 if DFSCTRN0 exists. This applies to AUTH and CHNG calls only. IMS calls this entry point regardless of the return code received from DFSCTRN0.

Whether you are operating in a RACF or non-RACF environment, DFSCTRN0 passes the return code directly to DFSCTSE0 in register 3.

Contents of registers on entry

On entry, the exit routine must save all registers using the save area provided. The registers contain the following:

Register	Contents
0	Address of the user ID from PST (PSTUSID)
1	Address of the password or zero For AUTH call, address of GENERIC class For TRAN call, address of TRAN class For FIELD call, address of FIELD class For DATABASE call, address of DATABASE class For SEGMENT call, address of SEGMENT class For OTHER call, address of OTHER class
2	Calling routine number as follows: **12 (X'0C')** DFSDLA30 for DFSCTSE0 only, CHNG call **32 (X'20')** DFSDLA30 for DFSCTSE0 only, AUTH call **12 (X'0C')** DFSDLA30 for DFSCTSE0 only, CHNG call **32 (X'20')** DFSDLA30 for DFSCTSE0 only, AUTH call
3	Return code from prior routines
4	For details of the format of this storage area, see the prolog in the sample routine (IMS.SDFSSMP; member name is DFSCTRN0).
7	Address of source CTB or zeros. **Recommendation:** Do not write an application that requires the contents of this register, because they vary depending on the type of call to the exit routine and the environment from which the call is made.
9	Address of PST.
10	Address of transaction code or resource name.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
15	Entry point of exit routine.

Contents of registers on exit

On return, all registers must be restored except for register 15, which must contain one of the return codes shown in the following table, to indicate the success or failure of the user's authorization to issue a AUTH or CHNG call.

Return code	Meaning
0	IMS accepts the CHNG call.
4	The resource is not protected.
8	The user is not authorized.
Positive	IMS rejects the CHNG call.
Negative	User is authorized. The negative value is the complemented address that points to user data provided by RACF (AUTH call).

Related reference

"Transaction Authorization exit routine (DFSCTRN0)" on page 307
The Transaction Authorization exit routine works with the Security Reverification exit routine (DFSCTSE0) and the Signon/off Security exit routine (DFSCSGN0) to check an individual user ID for authority to use a transaction.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Shared Printer exit routine (DFSSIML0)

The Shared Printer exit routine (DFSSIML0) decides whether a terminal that is unavailable can be automatically acquired by IMS or an AOI application program.

This information documents Product-sensitive Programming Interface and Associated Guidance Information provided by IMS.

Subsections:

About this routine

To acquire SLU 1, or 328X BSC/VTAM printers that are defined to IMS as shared, the IMS message router activates a Shared Printer exit routine. This is a routine that you write to decide whether a terminal that is unavailable can be automatically acquired by IMS or an AOI application program. The Shared Printer exit routine should return the name of the AOI application program.

A Shared Printer exit routine is not necessary to use shared printing. If no exit routine exists, the message router simulates a /OPN command when the terminal is defined as shared.

Attributes of the routine

The following table shows the attributes of the Shared Printer exit routine.

Table 103. Shared printer exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSSIML0.
Including the routine	The edit routine must be loaded into an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.
IMS callable services	To use IMS callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. Manually link this exit with DFSCSI00 to use callable services. No additional linking is required to use IMS callable services.
Sample routine location	IMS.ADFSSMPL (member name DFSSIML0).

Special considerations

If you decide to write a Shared Printer exit routine, here are some things you need to know:

- If the exit routine returns a bad return code, it is disabled and message DFS2084 is sent to the master terminal operator. A bad return code, in this case, is a return code of 8 when no transaction name is in the area pointed to by register 1 or when the transaction name returned is invalid. After the exit routine has been disabled, a return code of 0 is assumed. For the exit routine to be enabled, IMS must be restarted.
- The exit routine must not issue any waits, OS macros, or SVCs.
- The exit routine can examine output destination but cannot modify it.
- The exit routine should return the name of the AOI application program in the field provided by the message router.
- The exit routine receives control of the messages after they are queued.
- Because the exit routine runs in the IMS control region, your installation must maintain security. Installation procedures should not let an unauthorized routine be linked into the nucleus.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the area where the AOI transaction name is to be returned.
6	Address of CNT.
7	Address of CTB.
9	Address of CLB.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return code	Meaning
0	A SIMLOGON with the Q and RELRQ options is issued. This tells the other subsystem or VTAM application connected to the printer that IMS needs the printer.
4	No special processing is required. Normal processing continues.
8	The AOI transaction is activated. This transaction cannot be a conversational, Fast Path, remote, or password-protected transaction.

Related reference

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Signoff exit routine (DFSSGFX0)

Signoff exit routine (DFSSGFX0) performs processing that complements the Signon exit routine (DFSSGNX0). You can also use this exit routine to reset the significant status for terminals during user signoff.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

All attempts to sign off from ACF/VTAM terminals cause IMS to call this exit routine. The Signoff exit routine is also called if either RACF or the Signon/off Security exit routine (DFSCSGN0) fails a signon attempt.

Recommendation: Although the Signon exit routine and this exit routine are optional, if you include one, you should also include the other to perform any cleanup operations that are necessary.

The following table shows the attributes of the Signoff exit routine.

Table 104. Signoff exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSSGFX0.
Including the routine	If you want IMS to call the Signoff exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the current address ECB found at offset 0 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.ADFSSMPL (member name DFSSGFX0).

Extended Recovery Facility (XRF) considerations

Each time IMS calls the Signoff exit routine, the exit routine receives information on the XRF status of IMS. The exit routine can check this information and return the appropriate error message if necessary. IMS calls the exit routine if XRF tracking fails.

Resetting the significant status

You can use this exit to reset the significant status for a terminal in one of the following states:

Conversational
Exclusive
Test
Preset
MFS test
Full-function response
Fast Path response

Note: Test and preset states are nonrecoverable, so IMS resets the significant status automatically.

A parameter passed to the exit routine indicates the status of the terminal or ETO user at sign off. You can reset the status in the output parameters.

For conversation mode, IMS performs the equivalent of an /EXIT command for the conversation.

Restrictions

The Signoff exit routine cannot be used by LU 6.2 terminals.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

The following table lists the sign off parameters. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 105. Signoff exit parameter list

Offset (decimal)	Length	Description
+0	4	Current ECB address.
+4	4	SCD address.
+8	4	Address of the user table created by initialization user exit DFSINTX0 or zero, if none.
+12	4	Address of USERID associated with Sign Off.
+16	4	CLB address.
+20	4	Address of the STATUS_IN and STATUS_OUT vectors. The status vectors are mapped by the DFSSTCHK macro.

Contents of STATUS_IN

The input status vector is a 2-byte field that indicates the terminal's significant status when the exit routine is called. The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status as follows:

Value	Description
X'80'	Conversation
X'40'	Exclusive
X'20'	Test
X'10'	Preset
X'08'	MFS test
X'04'	Full-function response
X'02'	Fast Path response

Contents of STATUS_OUT

The output status vector is a two-byte field that indicates changes to the significant status made by the exit routine. IMS uses the contents of STATUS_OUT as an indicator to exit a conversation and reset significant status. The default for this field is zeros, indicating that no significant status is reset.

The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status to be reset as follows:

Value	Description
X'80'	Exit conversation
X'40'	Reset exclusive
X'20'	Reset test
X'10'	Reset preset
X'08'	Reset MFS test
X'04'	Reset full-function response
X'02'	Reset Fast Path response

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains one of the following return codes:

Return code	Meaning
0	Normal return.
Negative value	The specified user message is sent to the terminal signing off. This message can be used to trigger an AOI facility following a signoff operation.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“Initialization of IMS callable services (DFSCSII0)” on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

“IMS standard user exit parameter list” on page 5

Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Signon exit routine (DFSSGNX0)

IMS calls the Signon exit routine for signon processing if ETO=Y is specified as an execution parameter.

This topic contains Product-sensitive Programming Interface information.

This topic describes the Signon exit routine. All attempts to sign on to ACF/VTAM terminals if the Extended Terminal Option (ETO) feature is active cause IMS to call this exit routine. The Signon exit routine cannot be used by LU 6.2 terminals.

IMS calls the Signon exit routine before RACF validation (if requested) is performed and before the Signon/off Security exit routine (DFSCSGN0) is called. This exit routine contains logic and function that complement the Signon/off Security exit routine. Review your use of the Signon/off Security exit routine to determine if the function that it provides is necessary or conflicts with the Signon exit routine.

Related Reading:

- For more information on ETO and LU 6.2, see *IMS Version 15.2 Communications and Connections*.

Subsections:

About this routine

You can write the Signon exit routine to:

- Select the user descriptor (based on the user ID, node name, or DFSUSER) that you want IMS to reference when building the control block structure for the user that is signing on.
- Provide queue data for the user that is signing on. This could be data to override the queue data derived from the user descriptor. If the user descriptor is DFSUSER, the exit routine can also supply queue data to add additional LTERMs to the structure.
- Supply parameters that you want IMS to reference when building associated printer control block structures.
- Allow or disallow a signon attempt based on a maximum number of users, or according to any criteria that you specify.
- Specify, or override, autologoff parameter and autosignoff (ASOT) value.
- Override the default Status Recovery Mode for dynamic non-STSN terminals (terminals other than SLUP, FINANCE, and ISC).

For the latest version of DFSSGNX0, see the IMS.SDFSSMPL library; member name is DFSSGNX0. If you write your own Signon exit routine or modify the sample, you must include the portion of the sample exit routine (or the equivalent logic) that removes extraneous blank fields that RACF (if used) creates. (When the Signon exit routine is **not** included in the system, internal IMS logic removes these extraneous blank fields.) The sample exit routine also provides an example of associated printing.

When the Signon exit routine (DFSSGNX0) is not included in the system and the MFS formats for the DFS3649 message have not been modified, internal IMS logic removes these extraneous blank fields. If the MFS formats for the DFS3649 message have been modified, corresponding changes to the logic in the Signon exit routine that removes the extraneous blank fields might be necessary. This logic is included in the Signon exit routine so that adjustments can be made when changes are made to the DFS3649 MFS formats.

The Signon exit routine and the Destination Creation exit routine (DFSINSX0) are corequisite exit routines, under the following conditions. If you provide one exit routine to supply queue data for additional LTERMs, you must provide the other exit routine also. They both create the user control block structure and related LTERMs (including multiple LTERMs for a user): the Signon exit routine using the user ID and the

Destination Creation exit routine using an LTERM name. Both exit routines must have the same logic so that the structure created is identical, regardless of which exit routine created it.

You can use the Signoff exit routine (DFSSGFX0) to complement any processing that the Signon exit routine performs.

The following table shows the attributes of the Signon exit routine.

Table 106. Signon exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name the Signon exit routine DFSSGNX0.
Including the routine	If you want IMS to call the Signon exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.
IMS callable services	To use IMS callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the current address ECB found at offset 0 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.ADFSSMPL.

Assembling and loading the routine

A sample Signon exit routine is provided in IMS.SDFSSMPL. Alternatively, you can write your own exit routine. You can assemble the sample exit routine or one that you write (using the standard IMS macro and copy files), and include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL. If the Signon exit routine is included, IMS automatically loads it each time IMS is initialized if ETO=Y (after the Initialization exit routine, DFSINTX0, has changed the ETO= parameter).

If you want associated printing, be sure to specify the following when you assemble the sample exit routine:

```
&ASSOCPRT SETC 'YES'
```

This specification ensures that the associated printing sample code is generated.

User ID

The Signon exit routine informs the external subsystem of the user ID associated with the transaction input message. The user ID can be one of the following:

- The inputting LTERM name if the terminal user is not signed on
- The ID of the terminal user
- The RACF/user-authorized user ID associated with a non-message driven BMP or CPIC application
- The PSB name specified on the JOB statement

IMS determines the user ID in the following order.

For CPIC application:

1. RACF ID if the accessor environment element (ACEE) is cloned in the dependent region
2. PSTBUSER if the field does not contain binary zeros or blanks

3. PSTUSID if the field does not contain blanks
4. PSTSYMB0 if the field does not contain blanks
5. PDIRSYM

For a message driven BMP that has done a GU, or IFP that has done a GU, or MPP:

1. PSTUSID if the field does not contain blanks
2. PSTSYMB0 if the field does not contain blanks
3. PSTBUSER if the field does not contain binary zeros or blanks
4. PDIRSYM

For message driven BMP that has not done a GU or IFP that has not done a GU:

1. PSTBUSER if the field does not contain binary zeros or blanks
2. PDIRSYM

For non-message driven BMP:

1. PSTBUSER if the field does not contain binary zeros or blanks and the DFSDCxxx PROCLIB member specifies BMPUSID=USERID
2. PDIRSYM

When a dependent region connection is initially established, the Signon exit routine is activated before a thread is created by the Create Thread exit routine. All subsequent Signon requests result in the exit routine being activated after a thread is created. For example, Signon is activated for each message processed during a single scheduling, whether or not the messages are separated by commit processing.

Extended Recovery Facility (XRF) considerations

IMS calls the Signon exit routine in the XRF alternate system for a type 1 session with ETO. When IMS calls the exit routine in the alternate system, the exit routine is not allowed to change anything related to the terminal or user structures, including fields that the exit routine can normally change.

Each time IMS calls the Signon exit routine, the exit routine receives information on the XRF status of IMS.

Supporting associated printing

Associated printing is the ability to direct application printer output to a printer logical unit (LU) name. This LU name is provided at either logon or sign on time. If the Logon exit routine (DFSLGNX0) is written to detect LU names entered as logon user data, IMS passes these LU names to the Signon exit routine.

If you modify the DFS3649A MFS format to allow LU names to be entered as /SIGN ON user data, the Signon exit routine must be able to detect the LU names. If the user can enter LU names directly at sign on, the exit routine must determine the queue name that is allocated to each printer. There should be a unique relationship between the user ID and the queue name. The Signon exit routine passes the queue name to IMS, which creates the control block structure. An application program can use the same algorithm to determine the queue name when the application processes a transaction scheduled for a particular user ID.

The exit routine must insert a period (.) in the sign-on user verification string (UVS) after building the associated printer buffer.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)

Register	Contents
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

The following table lists the signon exit parameters. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 107. Signon exit parameter list

Offset (decimal)	Length	Description
+0	4	Current ECB address.
+4	4	SCD address.
+8	4	Address of the user table created by Initialization User Exit routine DFSINTX0 or zero, if none.
+12	4	Address of a buffer for use by your user exit to return user descriptor and queue data exit parameters. For additional details on the content and the format, refer to the prolog in the sample routine. Set to zero: • For a static terminal. • If processing on an XRF alternate system. • If processing `/SIGN ON ETO STSN` device. The USEQDATA DSECT is provided for parameter area mapping.
+16	4	Address of a buffer for use by your user exit to return associated printer exit parameters. For additional details on the content and the format, refer to the prolog in the sample routine. Set to zero, if processing on an XRF alternate system.
+20	4	Address of a parameter list created by one of the following: • Session initiation from LOGON data. • Input from the `/SIGN ON` command. For additional details on the content and the format, refer to the prolog in the sample routine. Set to zero, if processing XRF alternate.
+24	4	Address of a parameter list, which contains pointers to available user control block structures (SPQBs) and default autosignoff values. For additional details on the content and the format, refer to the prolog in the sample routine. Set to zero: • For a static terminal. • If processing on an XRF alternate system. • If processing /SIGN ON ETO STSN device.

Table 107. Signon exit parameter list (continued)

Offset (decimal)	Length	Description
+28	4	CLB address.
+32	4	Table of existing user structures. For additional details on the content and the format, refer to the prolog in the sample routine.
+36	4	Address of general Input/Output parameters. For additional details see DSECT DFSSGNXP macro for the format.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains one of the following return codes.

The content of register 15 will be ignored if processing is on an XRF alternate system.

Return code	Meaning
0	IMS continues SIGNON processing.
4	IMS rejects the SIGNON attempt. The SIGNON required message, DFS3649, is resent to the terminal with some added information indicating the reason for rejection.
Negative	The same as return code 4, but IMS sends the specified user message instead of DFS3649.

Related reference

"Signon/off Security exit routine (DFSCSGN0)" on page 287
Use the Signon/off Security exit routine (DFSCSGN0) to verify a user's ID and password.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

User descriptor selection

If the user control block structure already exists for the user that is signing on, IMS searches for the user structure and passes the addresses of the existing nodename structure along with the address of any existing user ID structure to the exit routine in the parameter list ESPQBTAB.

The exit routine can determine whether to use the user ID structure or the nodename structure by examining the passed structure without making an explicit IMS callable service routine call to find the nodename user structure.

If the exit routine chooses the nodename as the user structure name, the user ID is hashed to a non-SPQB user hash table.

If no user control block structure exists, you can select a user descriptor by using the USERD= keyword, write the Signon exit routine to select the user descriptor, or let IMS select a descriptor. The following figure shows the search order that IMS uses to select the user descriptor.

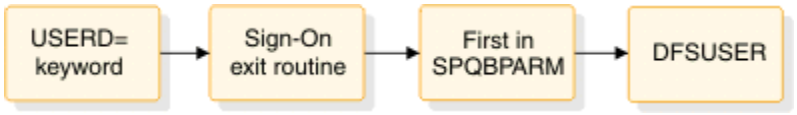


Figure 14. User descriptor selection order

You can use the USERD keyword (entering it as user data with the `/SIGN ON` command) to select the user descriptor. If your Signon exit routine does not choose a user descriptor, IMS uses the user descriptor requested by the USERD parameter.

The Signon exit routine is called with the parameter list SPQBPTRS, which contains the address of the USERD= keyword specified, and the addresses of the user ID descriptor, node name descriptor, and DFSUSER descriptor. The exit routine can choose among these descriptors by specifying the descriptor's address in the USEQUSED field of the USEQDATA DSECT. If the exit routine selects one of these user descriptors, IMS uses it to create the user control block structure. (A user descriptor that the exit routine specifies overrides any descriptor specified on the USERD= keyword.)

The exit routine can also create an arbitrary user structure name by specifying the name in the eight-byte USEQUSTN field in the USEQDATA parameter list. IMS creates the user structure with the name from this field and stores the returned name in the SPQB user hash table. Security is based on the original user ID that the user signed on with and is stored in the non-SPQB user hash table.

If no user descriptor is specified on the USERD= keyword and the Signon exit routine does not return the address of a user descriptor, IMS selects the first descriptor address that it finds in the SPQBPTRS among the user ID descriptor, node name descriptor, and the DFSUSER descriptor, respectively. IMS uses this descriptor to create the user control block structure.

If none of these methods returns a user descriptor, IMS uses DFSUSER to create the user structure. If no user descriptor can be found, including DFSUSER, IMS rejects the signon request.

Regardless of how the user descriptor is chosen, only DFSUSER or descriptors associated with the user ID or node name are valid. There is no user-based output security if the selected descriptor is the node name descriptor.

Providing queue (LTERM) data

Depending on the user descriptor selected, the exit routine can provide queue data. If the exit routine returns data that it was not authorized to return, IMS rejects the sign-on request.

Cases

Four cases describe what data the Signon exit routine (DFSSGNX0) can supply. The four cases are based on whether the user structure exists and whether DFSUSER or a non-DFSUSER descriptor is selected.

For the Signon exit routine, non-DFSUSER descriptors are descriptors based on the user ID or node name.

Table 108. Case numbers identifying what data DFSSGNX0 can provide

Descriptor	User structure exists	User structure does not exist
DFSUSER	Case 1	Case 2
Non-DFSUSER	Case 3	Case 4

Case 1

The Signon exit routine is called using the descriptor, DFSUSER, that was used to create the user control block structure. The exit routine can:

- Supply queue data (except LTERM names) to override data of the existing structure
- Provide data for additional LTERMs, if it supplies the data for the existing LTERMs first and in the order in which they are chained

IMS verifies the additional LTERMs that are specified (but are not in the existing user structure) against the LTERMs that already exist in the system. If an LTERM that is specified as an additional LTERM already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user that is signing on. If this is the only LTERM that the descriptor or the Signon exit routine specifies for this user, IMS rejects the signon attempt.

Case 2

If DFSUSER is selected and no user control block structure exists, the Signon exit routine:

- Can supply any queue data desired (including LTERM names)

If the exit routine does not provide queue data, one LTERM (named for the user ID) is created. If any queue data is passed, this default user ID LTERM is not created and must be specified in the queue data if it is desired.

IMS verifies the additional LTERMs that are specified against the LTERMs that already exist in the system. If an LTERM that is specified already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user that is signing on. If this is the only LTERM that the descriptor or exit routine specifies for this user, IMS rejects the sign-on attempt.

Case 3

The Signon exit routine is called with the same non-DFSUSER descriptor that was used to create the user control block structure (either the user ID or node name descriptor). The exit routine:

- Can supply any queue data (except LTERM names) to override data of the existing structure
- Cannot provide data for additional LTERMs

IMS verifies the LTERMs that are specified in the descriptor (but are not in the existing structure) against the LTERMs that already exist in the system. If an LTERM is specified in the descriptor but is not in the existing structure, IMS assumes that this LTERM has been assigned to a different user and deleted. The LTERM is given back to the user and is made part of the user structure of the user that is signing on.

Case 4

If a non-DFSUSER descriptor is selected and no user control block structure exists, the Sign On exit routine:

- Can supply queue data (except LTERM names) to override data that the descriptor provides
- Cannot provide data for additional LTERMs

IMS verifies the LTERMs specified in the descriptor against the LTERMs that already exist in the system. If an LTERM that is specified in the descriptor already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user that is signing on. If this is the only LTERM that the descriptor or exit routine specifies for this user, IMS rejects the signon attempt.

Related tasks

“User descriptor selection” on page 285
If the user control block structure already exists for the user that is signing on, IMS searches for the user structure and passes the addresses of the existing nodename structure along with the address of any existing user ID structure to the exit routine in the parameter list ESPQBTAB.

Signon/off Security exit routine (DFSCSGN0)

Use the Signon/off Security exit routine (DFSCSGN0) to verify a user's ID and password.

This topic contains Product-sensitive Programming Interface information.

This chapter describes the Signon/off Security exit routine. You can use this exit routine to verify a user's ID and password.

This exit routine can conflict with the Signon exit routine (DFSSGNX0).

Subsections:

About this routine

You can use the Signon/off Security exit routine with or without RACF to verify the user ID and password. IMS calls this exit routine after RACF /SIGN ON verification has been performed. If the /SIGN ON request is rejected by RACF, IMS does not call this exit routine. If the RACF option is not selected in the IMS system definition, you can use this exit routine to verify the user's identification and passwords at /SIGN ON time.

If ETO=Y is specified as an execution parameter, the Signon exit routine (DFSSGNX0) performs signon processing before IMS calls RACF or the Signon/off Security exit routine. If the Signon exit routine rejects the signon attempt, IMS does not call the Signon/off Security exit routine.

If shared queues are active and the security environment for a transaction is created on the back-end IMS subsystem, IMS does not call this exit routine.

The Signon/off Security exit routine should have access to a table of valid user IDs and the passwords associated with each ID. The exit routine should note successful /SIGN ONs to prevent additional attempts to perform the /SIGN ON function. When the /SIGN ON command is executed, the exit routine should mark that user ID as available for /SIGN ON. For logging purposes, the exit routine can also place information into the data portion of the user verification string that is passed to the exit.

If you plan to use the Signon exit routine, review how you use the Signon/off Security exit routine to determine if the function that this exit routine provides is necessary or might even conflict with the Signon exit routine.

Like both the Security Reverification exit routine (DFSCTSE0) and the Transaction Authorization exit routine (DFSCTRN0), the Signon/off Security exit routine does not need to be bound to the IMS nucleus, can run in 31-bit storage, and can share a work storage area using a standard technique.

The Signon/off Security exit routine is called during IMS initialization to give the exit routine the chance to acquire a work storage area. If storage is acquired, the exit routine passes the address back to IMS in register 2. Then, IMS passes the address to the DFSCTSE0, DFSCTRN0, and DFSCSGN0 security exit routines every time they are called.

If the Signon/off Security exit routine is linked in one of the STEPLIB or LINKLIST libraries, IMS loads the exit routine. There is no startup parameter to specify whether to load the routines. Message DFS1937I is issued when the Signon/off Security exit routine is loaded.

Signon/off Security exit routine is called after the Initialization exit routine (DFSINTX0) is called.

The following table shows the attributes of the Signon/off Security exit routine.

Table 109. Signon/off security exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCSGN0.
Including the routine	No special steps are required to include this routine.

Table 109. Signon/off security exit routine attributes (continued)

Attribute	Description
IMS callable services	To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for callable services. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.ADFSSMPL (member name DFSCSGN0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents

0	`/SIGN` function (ON or OFF):
	0
	`/SIGN ON`
	1
	`/SIGN OFF`
	2
	`/SIGN ON` in XRF alternate system.
	3
	`/SIGN OFF` in XRF alternate system.
	4
	IMS initialization. The exit can return an address that is passed to DFSCTRN0, DFSCTSE0, and DFSCSGN0.

1	Pointer to the variable-length user verification string, if the SIGN function is `/SIGN ON`. The string format is LLZZ (4 bytes), followed by the text, starting with the first character of the user ID. Address of the user ID if the SIGN function is `/SIGN ON` in an XRF environment. Insignificant if the SIGN function is `/SIGN OFF`.
7	Address of source CTB or zeros. **Recommendation:** Do not write an application that requires the contents of this register, because the contents of this register vary depending on the type of call to the exit routine and on the environment from which the call was made.
9	Address of ECB.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of registers on exit

On return to IMS, all registers must be restored except for register 15, which contains one of the following return codes:

Return code	Meaning
0	IMS accepts the `/SIGN ON`
4 (initialization only)	IMS stores the address returned by this exit routine and passes it to DFSCTRN0, DFSCTSE0, and DFSCSGN0.
Positive	IMS rejects the `/SIGN ON`. IMS sends message DFS2467 if sign-on is not required and message DFS3649 if sign-on is required. This return causes a "BY IMS EXIT" to be appended to the message to indicate that the exit routine caused the return code.
Negative	IMS rejects the `/SIGN ON` command and sends a user-defined message. The message number is complemented into a message number. This number must be less than -24, otherwise a DFS2467 message is sent instead. You must list the absolute value of this message number in the User Message Table, DFSCMTU0. **Exception:** The exit routine does not check this return code on return from RACF or during `/SIGN OFF` processing.

Related tasks

Extended Terminal Option (ETO) (Communications and Connections)

Related reference

"Signon exit routine (DFSSGNX0)" on page 281
IMS calls the Signon exit routine for signon processing if ETO=Y is specified as an execution parameter.

"Transaction Authorization exit routine (DFSCTRN0)" on page 307
The Transaction Authorization exit routine works with the Security Reverification exit routine (DFSCTSE0) and the Signon/off Security exit routine (DFSCSGN0) to check an individual user ID for authority to use a transaction.

"Security Reverification exit routine (DFSCTSE0)" on page 273
The Security Reverification exit routine (DFSCTSE0) allows you to reevaluate transaction authorization checking on the DL/I CHNG Call.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"Initialization of IMS callable services (DFSCSII0)" on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

"User Message table (DFSCMTU0)" on page 475
You can create your own messages and list them in the User Message table (DFSCMTU0).

Time-Controlled Operations (TCO) Communication Name Table (CNT) exit routine (DFSTCNT0)

The Time-Controlled Operations (TCO) Communication Name Table (CNT) controls which IMS LTERMs are allowed to load TCO scripts.

Subsections:

About this routine

The Time-Controlled Operations (TCO) Communication Name Table (CNT) exit routine gets control from the IMS Communication Analyzer module (DFSICIO0) whenever both of the following conditions are true:

- TCO is active.
- A message switch occurs for the DFSTCF LTERM.

The message switch acts as a load command from DFSTCF to load another TCO script. Use this exit routine to control which LTERMs are allowed to load TCO scripts.

The default exit routine immediately returns control to DFSICIO0, and you can load TCO scripts from any terminal.

This routine cannot be used in a DBCTL environment.

The following table shows the attributes of the TCO CNT exit routine.

Table 110. TCO CNT exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	If you are writing your own routine, you can give it any name. If you are using the IMS-supplied routine, use the name DFSTCNT0.

Binding	You should write, compile, and bind the routine as re-entrant (RENT). The following JCL is an example of binding a routine named MYEXIT to DFSICIO0.
	`//XIT JOB 1, MSGLEVEL=1`
	`//LINK EXEC PGM=IEWL, PARM=RENT`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL,DISP=SHR`
	`//INLIB DD DSN=IMS.OBJ,DISP=SHR`
	`//SYSLIN DD *`
	`INCLUDE INLIB(MYEXIT)`
	`INCLUDE INLIB(DFSICIO0)`
	`NAME MYEXIT(R)`
	`/*`
	In this example, IMS.SDFSRESL is an authorized library that contains all load modules. IMS.OBJ is a library that contains all object modules. The JCL in this example expects to find the object modules of the exit routine (MYEXIT) and the IMS Communication Analyzer module (DFSICIO0) in IMS.OBJ and places the result of the into IMS.SDFSRESL.
	After you've compiled and tested your routine (or if you are using the routine supplied by IMS), you must bind the exit routine with the TCO Language Interface module (DFSTDLI0).

IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSMPL (member name DFSTCNT0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	The buffer location of the input message segment after translation to EBCDIC and after IMS Basic Editing. The first two bytes of the buffer contain a binary message length. The third byte of the buffer is binary zeros. The binary count includes the 4-byte prefix. The fifth byte contains the first byte of message text.
7	Address of CTB.
9	Address of CLB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

Use the message segment in the buffer addressed by register 1 as input to the exit routine.

The exit routine must place the text of the edited message segment to be returned to IMS in the buffer addressed by register 1. If the input was processed by the IMS Basic Edit, this buffer is always 10 bytes greater than the 2-byte binary count at the beginning of the message segment. The length of the message segment can be expanded or reduced to any desired size. The format of the edited message segment in the buffer on return to IMS must be two bytes of binary count (LL), two bytes of binary zeros (ZZ), and edited text. The second two bytes (ZZ) should not be changed or edited. The LLZZ field is the first four bytes of the message segment.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes.

Return code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled and the user message identified by register 1 is sent to the terminal.

Register 1 contains the message number if register 15 contains a return code of 12; otherwise it is ignored. Any other value causes the message to be canceled and the terminal operator to be notified.

Time-Controlled Operations (TCO) exit routine (DFSTXIT0)

The TCO exit routine inserts messages in the message queue at a specific time for processing.

Subsections:

- "About this routine" on page 292
- "Communicating with IMS" on page 294

About this routine

The TCO exit routine inserts messages that are the commands, transactions, and message switches that you specify in the time schedule requests and message sets that make up a script member. The TCO exit routine passes any data found in columns 56 through 71 of the time schedule request to IMS to be processed.

You do not have to write your own exit routine. You can schedule predefined commands, transactions, and message switches at predefined times with DFSTXIT0, the TCO exit routine IMS supplies. If you do write your own, you can write it in COBOL or assembler.

Restriction: PL/I and C language exit routines are not supported. Cobol routines running under Language Environment for z/OS are not supported.

This routine cannot be used in a DBCTL environment.

The following table shows the attributes of the Time-Controlled Operations (TCO) exit routine.

Table 111. Time-Controlled Operations (TCO) exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	If you are writing your own routine, you can name it anything you want. If you are using the IMS-supplied routine, use the name DFSTXIT0.

Binding	You should write, compile, and bind the routine as serially reusable (REUS).
	The following JCL is an example of binding a routine named MYEXIT to DFSTDLI0.
	`//XIT JOB 1, MSGLEVEL=1`
	`//LINK EXEX PGM=IEWL, PARM=REUS`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL,DISP=SHR`
	`//INLIB DD DSN=IMS.OBJ,DISP=SHR`
	`//SYSLIN DD *`
	`INCLUDE INLIB(MYEXIT)`
	`INCLUDE INLIB(DFSTDLI0)`
	`NAME MYEXIT(R)`
	`/*`
	In this example, IMS.SDFSRESL is an authorized library that contains all load modules. IMS.OBJ is a library that contains all object modules. The JCL in this example expects to find the object modules of the exit routine (MYEXIT) and the TCO Language Interface module (DFSTDLI0) in IMS.OBJ and places the result of the bind into IMS.SDFSRESL.
	After you've compiled and tested your routine (or if you are using the routine supplied by IMS), you must bind the exit routine with the TCO Language Interface module (DFSTDLI0) and place them into IMS.SDFSRESL.

Including the routine	To load and execute the routine, it must be referred to in a time schedule request in the script member that is executing. **Related Reading:** For more information about time schedule requests and script members, see *IMS Version 15.2 Operations and Automation*. The following is an example of a time schedule request in a script member that wants the routine "MYEXIT" to be executed. `*TIME 1200 MYEXIT` • Columns 1-5 contain the Identification field. '*TIME' is in this field. • Columns 7-10 contain the initial dispatch time. In this example it is 12:00 p.m. • Columns 12-19 contain the name of the exit routine, left-justified and padded with blanks. The name in this example is 'MYEXIT'.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSRC (member name DFSTXIT0).

Communicating with IMS

IMS uses the entry and exit registers, and parameters to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of a parameter list that contains the address of the program communication block (PCB) used in the exit routine calls.
10	Reserved for TCO.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

The PCB

The program communication block (PCB) contains the actual scheduling time for the time-initiated message processing. It is in the PCBTIME field (PCB + 16). Under most circumstances, this is the same time as the time initiated request. In very busy systems, however, that actual scheduled time can differ from the schedule request. For example, if you request your exit routine to be scheduled at 12:01 and a busy system prevents it from being scheduled until 12:03, the PCB contains 12:03.

DL/I calls

The calls you can use in this exit routine are:

GU
Get the message that caused scheduling.

ISRT
Put a message segment into the queue for processing.

PURG
Terminate the prior segments as a message and insert the first segment on the next message (if an I/O area is provided).

GSCD
Get the address of the IMS system contents directory. The address is returned in the first word of the I/O area, which must begin on a word boundary.

The TCO exit routine calls the TCO Language Interface module (DFSTDLI0) to process these calls. You can call DFSTDLI0 or CBLTDLI0 (for COBOL) to process the call.

You must pass a parameter list with the call in standard DL/I format (for example, register 1 contains the address of a 2- or 3-word parameter list whose end is indicated by a X'80' in the high-order byte). The PURG call can have two or three parameters. The other calls require three parameters.

The parameter list consists of:

1. The call function
2. The address of the I/O PCB
3. The address of the I/O area (optional with PURG)

Status codes

A blank status code is returned to the exit routine after a successful call.

The following status codes can be returned to the exit routine after an unsuccessful call:

AB
The call didn't specify an I/O area.

AD
The function parameter on the call is invalid or is not supplied. The functions recognized by TCO are GU, ISRT, PURG, and GSCD.

AX
The I/O PCB name was invalid.

AZ
An ISRT or PURG call with an unacceptable message count was issued.

QC
There are no additional input messages to process for this time request.

QX
The ISRT or PURG call could not be processed because of insufficient storage.

Message formats

A GU call always retrieves a message in one of these formats:

- A 20-byte example as shown in the following figure.

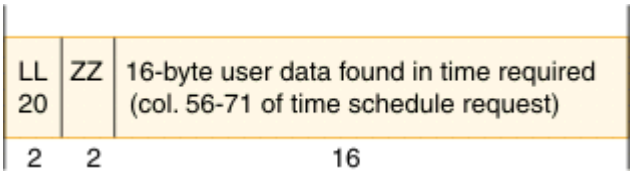


Figure 15. Format of 20-byte message example

- An 8-byte example as shown in the following figure.

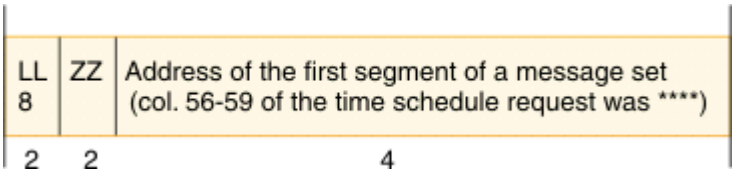


Figure 16. Format of 8-byte message example

- An example in which the address of a message set is retrieved as shown in the following figure.

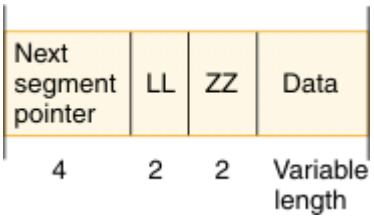


Figure 17. Format of message where the address of a message set is retrieved

The last message of the message set contains binary zeros in the "next segment" field.

If the message set is broken into individual messages and segments (by the use of a space and an S in column 72), this is shown in the ZZ field of each segment. The values are as follows:

Value	Meaning
0001	First segment of a message
0000	Middle segment of a message
0002	Last segment of a message
0003	First and last (only) segment of a message

TM and MSC Message Routing and Control User exit routine (DFSMSCE0)

The TM and MSC Message Routing and Control User exit routine (DFSMSCE0) provides maximum routing control for TM and MSC messages.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

The DFSMSCE0 user exit routine does the following:

- Eases TM and MSC coding and maintenance requirements, and reduces the number of exit modules.
- Supports a consistent set of routing capabilities across all of the exit entry points (or functions).

  This exit routine receives control for all User Type messages from:

  – Terminal/message input
  – MSC link input
  – Application program output

  In turn, the exit is allowed to affect the routing of most of these messages. Exceptions are cases where rerouting would violate IMS architecture or cause problems such as hung terminals or incorrect application program operation. For example, rerouting application program output messages to the I/O PCB is one of these exceptions (it is not allowed), or affinity routing of synchronous APPC/OTMA transaction messages to another IMS in a shared queues environment when the resource recovery service or APPC/OTMA enablement service is not set.

  For details on the routing capabilities for each exit entry point, see the user reroute flags MSTRFL2 (terminal), MSLRFL2 (MSC link), and MSPRFL2/MSPRFL3 (application) in the DFSMSCEP user parameter list mapping tables in Table 113 on page 302. Messages will be canceled or rerouted if you have set one of more of these flags in conjunction with the destination type.

  The user reroute request flags are MSTRFL2 (terminal), MSLRFL2 (MSC link), and MSPRFL2/MSPRFL3 (application). Setting one or more of these flags in conjunction with changing destination type fields, causes the message to be canceled or rerouted (see following note).

  See the DFSMSCE0 sample exit for examples of message routing.

  For affinity routing restrictions, see the topic "Managing APPC and OTMA messages in a sysplex environment" in *IMS Version 15.2 System Administration*.

  DFSMSCEP parameters that the exit can set or change to affect message routing are marked with a "U" or "B" as follows:

  I
  IMS SETS (EXIT MUST NOT CHANGE)

  U
  USER EXIT SETS

  B
  BOTH IMS/USER EXIT SET (OR CHANGE)

- Provides a common parameter list interface and linkage interface to the various entry points (or functions).
- Provides the ability to append an optional user prefix segment to TM and MSC messages which TM and MSC user exit routines can use to communicate and control user-customized routing needs.
- Provides new entry points:
  - Control at IMS initialization and termination
  - Control of messages in an MSC intermediate system
  - Application program inserts to a non-modifiable PCB

  All the entry points are optional, using a vector table that you code at the beginning of the common exit module.
- Logs routing errors and footprints in the message to indicate those exit routines that reroute the message.

The IMS initialization exit (DFSINTX0) cannot access MSC control blocks during IMS initialization, because the MSC control blocks are not built until restart. If the DFSINTX0 exit tries to access MSC control blocks, it will not find any. The MSC control blocks that cannot be found are LLB, LCB, LNB, and RCNT. The DFSMSCE0 user exit initialization entry point (which is called at IMS restart) and the other entry points can access those control blocks with FIND/SCAN control block callable services. See the prolog of that user exit for details and samples of those services.

Note: The DFSMSCE0 exit routine replaces the following exit routines:

- Input Message Routing exit routine (DFSNPRT0)
- Link Receive exit routine (DFSCMLR0/DFSCMLR1)
- Program Routing exit routine (DFSCMPR0)
- Terminal Routing exit routine (DFSCMTR0)

Sample IMS configurations

These samples describe four separate IMS configurations and the points where the DFSMSCE0 exit routine receives control during the flow of a transaction and response message.

Single IMS system

In a single IMS environment, the TR exit routine can receive control when a message is received from the terminal. The PR exit routine receives control when an application program issues a CHNG call to a modifiable PCB or on an ISRT call to a I/O or ALT PCB to insert a message, or a GU call to the I/O PCB.

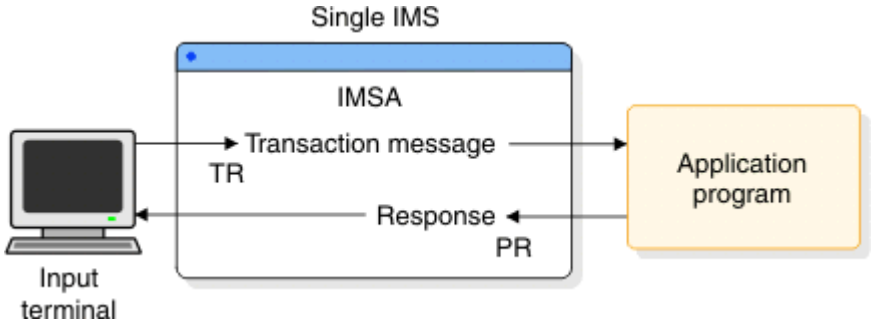


Figure 18. Single IMS system environment

Multiple Systems Coupling environment

In an MSC environment, the following occurs:

1. The TR exit routine receives control when a message is received from a terminal.
2. The PR exit routine receives control when the application program issues a CHNG call to a modifiable PCB or on an ISRT call to an I/O or ALT PCB to insert a message, or a GU call to the I/O PCB.

3. The LR exit routine receives control each time a message is received on an MSC link. The following figure shows the message flow when the transaction is received on the MSC link in IMSB (LR1) and on the MSC link in IMSC (LR2). In the response message flow, the LR exit receives control when the message arrives on the MSC link in IMSB (LR3) and when it arrives in IMSA (LR4).

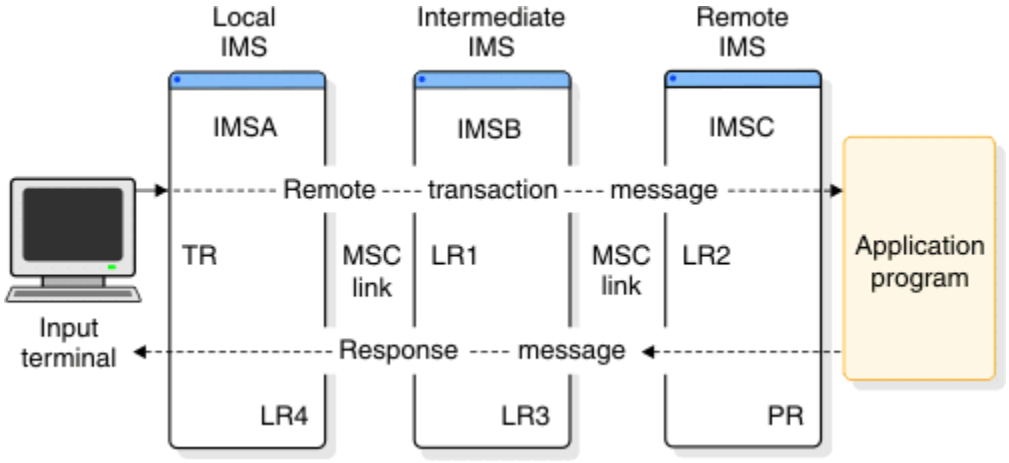


Figure 19. MSC environment

Shared-queues environment

A shared-queues environment is similar to a single-IMS environment. The TR exit routine receives control on the front-end IMS system, when the message is received from the terminal. The PR exit routine receives control on the back-end IMS system when the application program receives control and issues a CHNG or ISRT call to insert a message (PR).

The PR exit routine receives control when the application program issues a CHNG call to a modifiable PCB or on an ISRT call to the I/O or ALT PCB to insert a message, or a GU call to the I/O PCB.

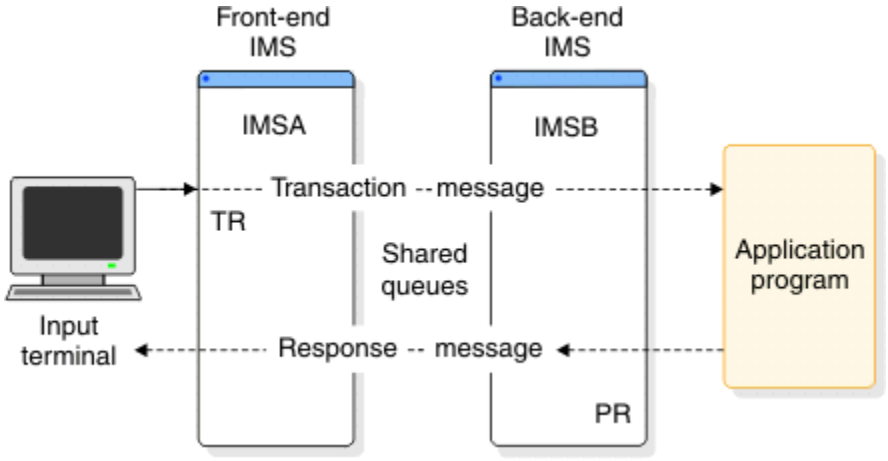


Figure 20. Shared-queues environment

Shared-queues MSC environment

The following occurs in a shared-queues MSC environment:

1. The TR exit routine receives control in the front-end IMS system when the transaction message is received from the terminal.
2. The PR exit routine receives control in the front-end IMS (PR1), the back-end IMS (PR2), or the remote IMS (PR3) systems when the application program receives control and issues a CHNG call to a modifiable PCB or on an ISRT call to the I/O or ALT PCB to insert a message, or a GU call to the I/O PCB.

3. The LR exit routine receives control in the remote IMS system when the transaction message is received on the MSC link (LR1), and then in the back-end IMS system when the response message is received on the MSC link (LR2).

In a shared-queues environment, two additional levels of affinity routing are available for messages destined to a transaction. One level requests the transaction message to be routed locally on the current IMS system. This is referred to as local affinity. The other level requests the transaction message to be routed to the back-end IMS system. This is referred to as back-end affinity. Affinity routing is available in the terminal, link receive, and program routing entry points.

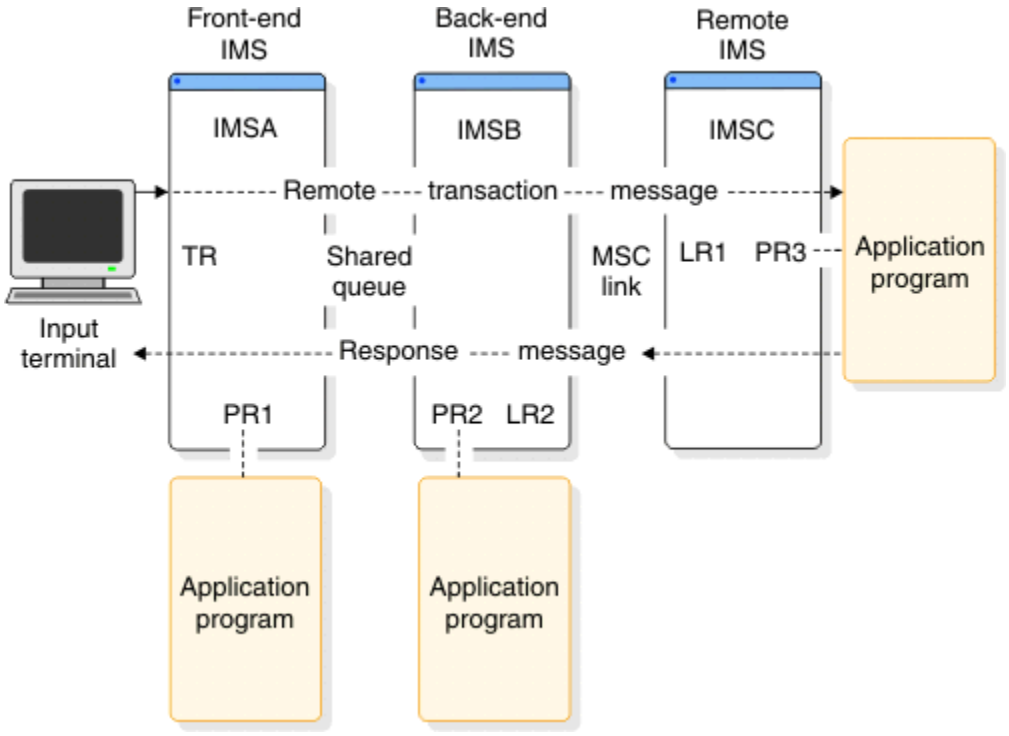


Figure 21. Shared-queues MSC environment

Defining entry points

You can define the entry points and conditions for IMS to call the DFSMSCE0 exit routine by coding the user vector table macro (DFSMSCVT). In the front of the module, code the VECTOR=MSCVTABLE parameter in the DFSMSCSV macro to point to the resulting vector table. The DFSMSCVT macro supports 12 entry points that you can specify to select those conditions for which the exit routine is called (2 for IMS initialization and termination, and 10 entry points in the flow of TM message processing).

The entry points enable:

- Rerouting a message to a different destination name or a different remote IMS in a MSC system.
- Requesting transaction affinity processing (processing messages in a specific IMS) in a shared queues IMSplex system by requesting that the message process locally in the current IMS or in a different back-end IMS.
- Rejecting the message

The DFSMSCE0 user exit routine can change the routing of a message by setting flags and fields in the user parameter list that IMS passes to the exit routine. This parameter list is mapped by the DFSMSCEP macro, and then returned to IMS. The parameter list contains:

- Fields and flags to indicate IMS conditions, such as MSC or shared-queues system definition
- Information regarding the message, such as source and destination names and MSC system identifiers (SYSIDs) for routing control

Some of the information in the parameter list is for reference only, while other information can be changed to affect the rerouting of the message. See the DFSMSCEP macro, described in Table 113 on page 302 through Table 118 on page 305, for more information.

At any of the user exit entry points (other than the initialization or termination entry points), the exit routine can request a user prefix segment to be added to the message. If a user prefix is already obtained for this message by a previous call to the exit routine, IMS passes the address of the user prefix to the exit routine. The exit routine can reference or change the user prefix, but cannot delete it or change its length. This prefix can contain user routing information that can be passed to the other routing exit entry points to be used to reroute the message. After the user prefix is obtained, it remains appended to the message and is logged with the message (for example, a type 01 or type 03 message log record is mapped by the QLOGMSGP macro).

For each routing request, the user exit routine is passed a 512–byte work area that is initialized to zeros and that the user exit routine can use as a work area, such as for creating a user prefix.

No IMS System Definition changes are needed to invoke the DFSMSCE0 exit routine, and MSC does not need to be available; however, several of the routing functions are only available for MSC messages. The DFSMSCE0 exit routine is loaded at IMS initialization, provided that the load module is link edited into IMS.SDFSRESL or a user library concatenated to IMS.SDFSRESL.

Authorization checking

The exit call during link receive processing controls the level of authorization checking. The level of authorization is controlled by the field MSLRFL3 of the parameter list during link receive. IMS sets one of the flags in MSLRFL3 when calling the link receive entry points to indicate which level of security checking is active. If the message is a local transaction message, resetting or changing this flag will override the level of security to be performed for this message. Flag MSLRFL1 can be tested to determine if the message is a local transaction. The following parameters in the MSLRFL3 field specify the level of authorization:

MSLR3MSN
Authorization by MSNAME. The accessor environment element (ACEE) dynamically created for first authorization, then reused.

The specification of MSLR3MSN causes the security environment based on the MSNAME to be built the first time it is needed for an authorization check. Thereafter, the environment is saved and is reused for subsequent checking.

MSLR3CTL
Authorization by CTL address space security. The specification of MSLR3CTL uses the security environment of the CTL address space that already exists.

MSLR3USR
Authorization by user ID of input terminal. ACEE dynamically created and deleted for each authorization.

The specification of MSLR3USR causes the security environment based on the user ID of the input terminal (that entered the transaction) to be built each time it is needed for an authorization check.

MSLR3XIT
Authorization by user exit (DFSCTRN0). MSLR3XIT can be specified by itself, or with either MSLR3MSN, MSLR3CTL, or MSLR3USR. The specification of MSLR3XIT causes DFSCTRN0 or DFSCTSE0 to be called, if they exist.

MSLR3NON
No security authorization checking.

MSLR3NON can only be specified without any of the other four options. The specification of MSLR3NON bypasses all security checking, and allows the use of the transaction destination.

MSLR3MSN, MSLR3CTL, and MSLR3USR are mutually exclusive. The use of MSLR3MSN, MSLR3CTL, or MSLR3USR causes RACF (or an equivalent product) to be called for authorization of the use of the transaction destination.

On entry, the MSLRFL3 field contains the system default value from MSCSEC=(,xxx) in the DFSDCxxx PROCLIB member. The exit can then override the system default, or leave it as is.

Attributes of the routine

The TM and MSC Message Routing and Control user exit routine must be written as reentrant. The exit routine receives control while running in a 31-bit addressing mode, and must return control in that mode. The exit routine is called in TASK mode, with no locks held, and can be in cross memory, non_AR mode.

The following table shows the attributes of the TM and MSC Message Routing and Control User exit routine.

Table 112. TM and MSC message routing and control user exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	Must be named DFSMSCE0.
Binding	This exit routine must be reentrant. The sample exit routine is a default routine. If you write your own exit routine, you must bind it with the IMS control region SDFSRESL. Link edit stand alone, NAME/ENTRY = DFSMSCE0, RMODE=ANY, AMODE=31, and Reentrant (RENT). Program routing entry points (DFSMSCVT ENTRYP=PRCHNG, PRISRT) execute in cross-memory mode under the dependent region TCB. All other entry points execute under the control region TCB.
Including the routine	IMS loads and initializes the exit if found in IMS.SDFSRESL or concatenated library. The module has 12 possible entry points selectable by the ENTRYP parameter of the DFSMSCVT macro coded in the module (see sample DFSMSCE0).
IMS callable services	To use callable services with this exit routine, it must be given a callable services token by IMS when it is given control. To determine if you can use callable services, check the value of the SXPLATOK field in the "IMS standard user exit parameter list" on page 5: • If the value of SXPLATOK is zero, you cannot use callable services with this exit routine. • If the value of SXPLATOK is non-zero, the callable services token is included and you can use callable services with this routine. Use the 256-byte work area addressed by the SXPLAWRK field to call DFSCSIF0.

Table 112. TM and MSC message routing and control user exit routine attributes (continued)

Attribute	Description
Sample routine location	**Recommendation:** Use the sample DFSMSCE0 exit routine that is shipped in IMS.ADFSSMPL and tailor it when first coding the user exit routine. This sample contains examples of the following: • Routing messages, using all the supported routing options (by setting the appropriate flags and fields in the DFSMSCEP area). • Canceling messages. • Using the DFSMSCVT (entry vector table) macro and all 12 entry points. • Using the DFSMSCSV (save) macro to set up the entry environment. • Using the DFSMSCLV (leave) macro to return to IMS. • Chaining and using the 6 save sets that are passed to the exit routine. • Using the 512–byte work area to build a user prefix and requesting that IMS obtain a prefix buffer to build a prefix. • Storing information in the user prefix

Communicating with IMS

This section provides information about how to communicate with IMS using the DFSMSCE0 user exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of the “IMS standard user exit parameter list” on page 5
R13	Address of save area
R14	Return address
R15	Address of entry point

This exit routine uses the Version 6 standard exit parameter list. The address of the work area that is passed to this exit routine in SXPLAWRK can be different each time that this exit routine is called.

The DFSMSCE0 user parameter list and field definitions are mapped by the DFSMSCEP macro.

Table 113. Main user exit parameter list mapped by the DFSMSCEP macro

Field	Offset	Length	Description
MSCEIMID	00	8	IMSID of this IMS
MSCEIMSR	08	1	Source IMS release number
MSCEIMSL	09	1	Source IMS mod level
MSCEPLVER	0A	2	DFSMSCEP parameter list version (current version=0004)
MSCEFL1	0C	1	Main flag 1
MSCEFL2	0D	1	Main flag 2
MSCEFL3	0E	1	Main flag 3

Table 113. Main user exit parameter list mapped by the DFSMSCEP macro (continued)

Field	Offset	Length	Description
MSCEFL4	0F	1	Main flag 4
MSCEECB	10	4	Address of ECB
MSCESCD	14	4	Address of SCD
MSCESIDT	18	4	Address of SID_Table
MSCESEG	1C	4	Address of MSG_Segment
MSCEUPR	20	4	Address of User_PFX_Seg
MSCEIPR	24	4	Address of IMS_PFX_Seg
MSCEUPRL	28	2	User_PFX_Len (halfword)
MSCEIPRL	2C	2	IMS_PFX_Len (halfword)
MSCESSET	2E	4	Address of Save_sets
MSCEMSEB	30	4	Address of DFSMSCEB
	34	4	Reserved
MSCEUSID	38	8	User ID
MSCEGRPN	40	8	Group name
MSCEUSII	48	1	User ID indicator
	49	3	Reserved
MSCEAFIN	4C	8	IMSID to route message for shared queues affinity routing
	54	20	Reserved
	68		End main parameters

The initialization entry parameter list and field definitions are mapped by the DFSMSCEP macro.

Table 114. Initialization entry parameters for user exit parameter list mapped by the DFSMSCEP macro

Field	Offset	Length	Description
MSINFL1	68	1	Initialization flag1
MSINFL2	69	1	Initialization flag2
MSINFL3	6A	1	Initialization flag3
MSINFL4	6B	1	Initialization flag4
	6C	12	Reserved
	78		End of IMS initialization parameters

The termination entry parameter list and field definitions are mapped by the DFSMSCEP macro.

Table 115. Termination entry parameters for user exit parameter list mapped by the DFSMSCEP macro

Field	Offset	Length	Description
MSTEFL1	68	1	Termination flag1
MSTEFL2	69	1	Termination flag2

Table 115. Termination entry parameters for user exit parameter list mapped by the DFSMSCEP macro (continued)

Field	Offset	Length	Description
MSTEFL3	6A	1	Termination flag3
MSTEFL4	6B	1	Termination flag4
	6C	12	Reserved
	78		End of IMS termination parameters

The terminal routing parameter list and field definitions are mapped by the DFSMSCEP macro.

Table 116. Terminal routing parameters for user exit parameter list mapped by the DFSMSCEP macro

Field	Offset	Length	Description
MSTRFL1	68	1	XL1 TR flag1
MSTRFL2	69	1	XL1 TR flag2
MSTRFL3	6A	1	XL1 TR flag3
MSTRFL4	6B	1	XL1 TR flag4
MSTRDEST	6C	8	DEST_NAME
MSTRSRCE	74	8	SRCE_NAME
MSTRLUNM	7C	4	LU_NAME
MSTRMSGR	80	4	APPC_WORK
MSTRDMSN	84	8	MSNAME
MSTRDSID	8C	2	Dest_SID
MSTRKEY	8E	2	MSG_KEY
MSTRLTMN	90	8	OTMA destination override name
	98	16	Reserved
	A8		End of terminal routing parameters

The link receive parameter list and field definitions are mapped by the DFSMSCEP macro.

Table 117. Link receive routing parameters for user exit parameter list mapped by the DFSMSCEP macro

Field	Offset	Length	Description
MSLRFL1	68	1	Link receive flag1
MSLRFL2	69	1	Link receive flag2
MSLRFL3	6A	1	Link receive flag3
MSLRFL4	6B	1	Link receive flag4
MSLRDEST	6C	8	DEST_NAME
MSLRSRCE	74	8	SRCE_NAME
MSLRDMSN	7C	8	DST_MSNAME
MSLRDSID	84	2	DEST_SID
MSLRSMSN	86	8	SRC_MSNAME

Table 117. Link receive routing parameters for user exit parameter list mapped by the DFSMSCEP macro (continued)

Field	Offset	Length	Description
MSLRSSID	8E	2	Source_SID
MSLRKEY	90	2	MSG_KEY
	92	22	Reserved
	A8		End of link receive routing parameters

The program routing parameter list and field definitions are mapped by the DFSMSCEP macro.

Table 118. Program routing parameters for user exit parameter list mapped by the DFSMSCEP macro

Field	Offset	Length	Description
MSPRFL1	68	1	Program routing flag1
MSPRFL2	69	1	Program routing flag2
MSPRFL3	6A	1	Program routing flag3
MSPRFL4	6B	1	Program routing flag4
MSPRDEST	6C	8	DEST_NAME
MSPRSRCE	74	8	SRCE_NAME
MSPRDMSN	7C	8	DST_MSNAME
MSPRDSID	84	2	DEST_SID
MSPRDMSN	86	8	DEST_MSNAME
MSPRSSID	8E	2	Source_SID
MSPRSTAT	90	2	Status_Code
	92	22	Reserved
	A8		End of program routing parameters

The DFSMSCE0 exit routine is called with one caller save area in R13. Field MSCESSET in DFSMSCEP points to six preformatted save sets for the exit routine's use. The routine (INITSAV) in the sample exit routine (DFSMSCE0) chains these save sets to the caller save set and moves R13 to the first save set in MSCESSET. This allows the DFSMSCE0 exit routine to call other routines and to pass a save set chain. When DFSMSCE0 returns to IMS, the DFSMSCLV macro (Linkage=Yes) returns to the caller save set and restores registers.

Callable services

Storage services and control block services can be performed by invoking IMS callable services. This exit routine can use callable services with the ECB passed at MSCEECB of the user exit PARMLIST.

This exit routine can use IMS Callable Storage Services. This exit routine is defined to IMS as an IMS standard user exit. Exit routines that are defined to IMS receive the callable services token in the standard exit parameter list. This exit routine does not need to issue an initialization call (DFSCSII0) to use IMS callable services.

The exit routine receives control at the following points: the Terminal Routing (TR) call, the Link Receive (LR) call, and the Program Routing (PR) call. In each situation, if the DFSMSCE0 user exit routine is called (based on the DFSMSCVT vector entry) and obtains a user prefix, IMS attaches the prefix to the message and passes it on to other DFSMSCE0 entry points.

For each entry point parameter selected by the DFSMSCVT macro, the exit routine must provide a label for the entry point, as shown in the following table.

Table 119. Labels for entry point parameters selected by the DFSMSCVT macro

Parameter	Label	Function/when called
1. INIT	IMS_INITIALIZATION	IMS initialization
2. TERM	IMS_TERMINATION	IMS termination
3. TRBTAM	TERMINAL_ROUTING_BTAMS	System console message
4. TRVTAM	TERMINAL_ROUTING_VTAM	VTAM messages
5. TRAPPC	TERMINAL_ROUTING_APPC	APPC messages
6. TROTMA	TERMINAL_ROUTING_OTMA	OTMA messages
7. LRTRAN	LINK_RECEIVE_LOCAL_TRANSACTION	Local tran messages
8. LRLTERM	LINK_RECEIVE_LOCAL_LTERM	Local LTERM messages
9. LRDIR	LINK_RECEIVE_LOCAL_DIRECT_ROUTING	Local DIR RTE messages
10. LRINT	LINK_RECEIVE_INTERMEDIATE	Intermediate messages
11. PRCHNG	PROGRAM_ROUTING_CHNG_CALL	Application program CHNG call
12. PRISRT	PROGRAM_ROUTING_ISRT_CALL	First message segment ISRT call
13. PRGU	PROGRAM_ROUTING_ISRT_CALL	Application program issued GU call

The DFSMSCVT macro parameters listed in the preceding table have the following characteristics:

INIT entry point
Receives control at IMS initialization, immediately after the exit routine is loaded.

TERM entry point
Receives control at IMS termination when IMS is shutting down. The INIT and TERM entry points are not associated with a message.

The next 4 entry points are for the Link Receive (LR) user exit routine:

LRTRAN
Receives control when a message is received on an MSC link, and the destination is a local transaction in the received system.

LRLTERM
Receives control when a message is received on an MSC link, and the destination is a local LTERM in the received system.

LRDIR
Receives control when a direct-routed message is received for the local IMS system. The destination can be an LTERM or a transaction. Direct-routed messages are created by an application program running in a remote MSC system that inserts messages using directed routing (in other words, inserts messages to a PCB MSNAME destination).

LRINT
Receives control for any message received on an intermediate IMS system (in other words, a message received on an MSC link that is destined to another remote MSC system). This includes intermediate messages that are inserted by a remote IMS system using directed routing.

The next 2 entry points are for the Program Routing (PR) user exit routine:

PRCHNG
Receives control when an application program issues a CHNG call to a modifiable PCB.

PRISRT
Receives control when an application program issues the first ISRT call (first segment) to a modifiable PCB, non-modifiable PCB, or I/O PCB.

PRGU
Receives control when an application program issues a GU call to a I/O PCB. The exit may request or update a user prefix but no message routing is supported.

Using user prefixes

Messages contain a variety of prefixes that IMS uses to route and process the message. These prefixes are mapped by the QLOGMSGP macro, and are in front of the message, before the user data segments. These prefixes are for internal IMS use. DFSMSCE0 can add a user prefix to this message. This prefix is mapped by the DFSMSCUP macro. The exit routine can build this prefix in one of two ways:

- Test the field MSCEUPR in DFSMSCEP for zero to see if a user prefix already exists. If not obtained (zero), build a prefix in the 512–byte work area by addressing some area in the work area that is large enough to hold the prefix. Set bytes 0 and 1 to the prefix length (5 to 512 bytes), storing the address back in MSCEUPR. The exit routine can then alter the user data portion of the prefix (bytes 4 to 512). When the exit routine returns control to IMS, IMS sets the prefix code (byte 2 = 8E) and the reserved flag (byte 3) and copies the prefix to the message.
- Test the field MSCEUPR in DFSMSCEP for zero to see if a user prefix already exists. If not obtained (zero), set flag MSCE2UPR=1 and field MSCEUPRL to the length of the requested prefix (5 to 512 bytes) and return to IMS. IMS obtains storage that is large enough for the user prefix and stores the address in MSCEUPR, resets flag MSCE2UPR, and returns control to the exit routine. The exit routine can then alter the user data portion of the prefix (bytes 4 to 512). When the exit routine returns control to IMS, IMS sets the prefix code (byte 2 = 8E) and the reserved flag (byte 3) and copies the prefix to the message, and then frees the original prefix storage.

Note: If the user prefix is obtained for the DFSMSCE0 exit, the size of that prefix should be considered along with the accumulated size of the other prefix items when calculating the record lengths for the short and long message queue records.

Related reading: For more information on MSGQUEUE macro message prefix sizes for each supported IMS release, see *IMS Version 15.2 System Definition*.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“IMS standard user exit parameter list” on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Transaction Authorization exit routine (DFSCTRN0)

The Transaction Authorization exit routine works with the Security Reverification exit routine (DFSCTSE0) and the Signon/off Security exit routine (DFSCSGN0) to check an individual user ID for authority to use a transaction.

This information documents Product-sensitive Programming Interface and Associated Guidance Information provided by IMS.

Subsections:

About this routine

This exit routine can be used with or without RACF to verify that the user's ID is authorized to run a transaction. If the RACF option is selected and the Transaction Authorization exit routine is loaded, the exit is activated after RACF verifies the transaction. If the transaction request is rejected by RACF, the exit is not called. If the RACF option is not selected in the IMS system definition, this exit routine can be used to verify the user's authorization and the password, if required, for that transaction.

Attention: Changing RCF=N to RCF=R requires a cold start of the IMS control region.

The exit routine should have access to a table of valid user IDs, and the passwords and transactions associated with each valid user ID.

If you want to generate your own messages for the routine, you need to make the message number negative in register 15 to issue a specific message, and you need to list the absolute value of this message number in the User Message Table, DFSCMTU0. For details, see "User Message table (DFSCMTU0)" on page 475.

If you do not list this message in the User Message Table, message DFS060I is issued instead of the message you wanted to send.

The IMS security exit routines do not need to be bound to the IMS nucleus, can run in 31-bit storage, and can share a work storage area. The following security exit routines now have these attributes:

- Signon/off security exit routine (DFSCSGN0)

  DFSCSGN0 is called during IMS initialization to give the exit routine the chance to acquire a work storage area. The exit routine passes the address back to IMS. Then, IMS passes the address to the other security exit routines every time they are called.
- Security Reverification exit routine (DFSCTSE0)
- Transaction Authorization exit routine (DFSCTRN0)

If the security exit routines are linked in one of the STEPLIB or LINKLIST libraries, IMS loads the exit routine. There is no startup parameter to specify whether to load the routines. Message DFS1937I is issued for every exit routine that is loaded into 31-bit storage.

If distributed network security credentials, including a network user ID and a network session ID, are included in the security-data section of the OTMA message prefix, the address of the security credentials in the OTMA message prefix is included in the storage area of the Transaction Authorization exit routine.

The following table shows the attributes of the Transaction Authorization exit routine.

Table 120. Transaction authorization exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCTRN0.

Table 120. Transaction authorization exit routine attributes (continued)

Attribute	Description
Binding	The Security Reverification exit routine (DFSCTSE0) can be bound to DFSCTRN0 or coded as an explicit part of DFSCTRN0. If you code this entry point, it should have access to a table of valid user IDs, passwords, and transactions associated with each valid user ID, or contain some algorithm to derive this authorization information. For addressability, this table should reside in this module, in the / SIGN ON exit (DFSCSGN0), or in the IMS nucleus. The security exit routines can be bound separately. If the security exit routines are linked in one of the STEPLIB or LINKLIST libraries, IMS loads the exit routine. There is no startup parameter to specify whether to load the routines. IMS issues message DFS1937I each time a DFSCSGN0, DFSCTRN0, or DFSCTSE0 exit routine is loaded. If the exit routines cannot be linked separately or cannot use a common work area, they must be linked in the following manner: • If the CSECT of DFSCTSE0 is part of DFSCTRN0 source, DFSCTSE0 must be linked as an ALIAS of DFSCTRN0. • If virtual address spaces are used to exchange data between DFSCSGN0, DFSCTRN0, and DFSCTSE0, then DFSCTSE0 and DFSCSGN0 must be linked as ALIASs of DFSCTRN0.
Including the routine	Include the exit routine by linking it in either the STEPLIB or LINKLST library. IMS detects and loads it automatically. You do not need to specify any system definition or startup parameters. IMS confirms that the exit routine is loaded by issuing a DFS1937I message.
IMS callable services	To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB in register 9 for the DFSCSII0 call. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.ADFSSMPL (member name DFSCTRN0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Register contents is dependent on what is processed: • To process a deferred program-to-program switch (R2 = 8), or DL/I CHNG call (R2 = C), then R0 = pointer to user ID (PSTUSID). • To process receipt of a transaction received on an MSC link from a remote IMS system (R2 = 4), then R0 = pointer to user ID in the security prefix of the message. This exit routine is called when R2 = 4 depending on the MSCSEC par\ameter in DFSDCxxx and on the MSLRFL3 response in the DFSMSCE0 parameter list for Link Receive. For more information on the MSCSEC parameter, see *IMS Version 15.2 System Definition*.

Register	Contents
1	Address of the password or password phrase: **Note:** To determine the length of the password or password phrase, refer to the parameter list storage area passed in Register 3. • For AUTH call, address of GENERIC class • For TRAN call, address of TRAN class • For FIELD call, address of FIELD class • For DATABASE call, address of DATABASE class • For SEGMENT call, address of SEGMENT class • For OTHER call, address of OTHER class
2	Calling routine number: **Number** **Name** **X'0'** Transaction input from terminal **X'4'** Transaction from remote MSC system **X'8'** Deferred conversation program-to-program switch **X'C'** CHNG DL/I call **X'10'** /SET command **X'14'** /LOCK command **X'1C'** /RELEASE command **X'20'** AUTH call **X'24'** LU 6.2 AUTH call **X'28'** Transaction input from OTMA **X'2C'** /LOCK and /UNLOCK transaction **X'30'** /LOCK and /UNLOCK program **X'34'** /LOCK and /UNLOCK database **X'38'** /LOCK and /UNLOCK LTERM **X'3C'** Remote deferred program switch
3	Address of storage area. For details of the format of this storage area, see the prolog in the sample routine (IMS.ADFSSRC; member name is DFSCTRN0).

Register	Contents
7	Address of source CTB or zeros. **Recommendation:** Do not write an application that requires the content of this register, because they vary depending on the type of call to the exit routine and the environment from which the call is made.
9	Address of the ITASK control block: **If Register 2 is** **Address of Register 9 will be** **X'0'** CLB **X'4'** LLB **X'8'** PST **X'C'** PST **X'10'** CLB **X'14'** CLB **X'1C'** CLB **X'20'** PST **X'24'** CLB **X'28'** PST **X'2C'** CLB **X'30'** CLB **X'34'** CLB **X'38'** CLB **X'3C'** CLB
10	Address of transaction code or resource name.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of registers on exit

On return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes to indicate the success or failure of the user's authorization to a transaction.

Return code	Meaning
0	Accept the transaction.
4	The resource is not protected.
8	The user is not authorized.
Positive	Reject the transaction and send DFS2469 message with register 15 halfword contents as a subcode if the transaction is entered from a terminal. The IMS system translates the subcode of message DFS2469 as follows: **Subcode** **Meaning** **08** Transaction not authorized (user is not authorized). **12** RACF is not active. **16** Invalid exit return code. **36** No password (password reverification is required, but no password was supplied). **40** Wrong password (password reverification failed). **Others** IMS exit CD (subcode generated by IMS exit).
Negative	For Resource Authorization: User is authorized. The negative value is the complemented address that points to user data provided by RACF (AUTH call).
Negative	For Transaction Authorization: Reject the transaction and send a user-defined message number, if appropriate, to the user. If the calling routine is DFSCON10 or DFSDLA30, no message is sent, but an A4 status code is passed to the application program. The message number passed must be less than -24.

Related reference

“Security Reverification exit routine (DFSCTSE0)” on page 273
The Security Reverification exit routine (DFSCTSE0) allows you to reevaluate transaction authorization checking on the DL/I CHNG Call.

“Signon/off Security exit routine (DFSCSGN0)” on page 287
Use the Signon/off Security exit routine (DFSCSGN0) to verify a user's ID and password.

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“Initialization of IMS callable services (DFSCSII0)” on page 16

Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Transaction Code (Input) edit routine (DFSCSMB0)

Use the Transaction Code (Input) edit routine (DFSCSMB0) to define IMS transactions.

This topic contains Product-sensitive Programming Interface information.

This topic describes the Transaction Code (Input) Edit routine.

Subsections:

About this routine

Messages that are entered for the transaction are passed to the Transaction Code Input edit routine before they are queued for scheduling. This sequence enables you to edit input messages before they are placed on the message queues. The Transaction Code Input edit routine is called in addition to the IMS Basic Edit routine or MFS (Message Format Service) editing. The message is passed to the input edit routine before it is translated to uppercase characters.

Transaction code input edit routines can be defined to IMS either through the system definition process or dynamically by using a DRD command. You can define up to 255 different Transaction Code Input edit routines for each IMS.

You can define a Transaction Code Input edit routine during IMS initialization by using the EDIT parameter on the TRANSACT macro. The edit routine must be included in one of the IMS.SDFSRESL concatenated data sets.

You can dynamically define a Transaction Code Input edit routine by using DRD commands. The EDITRTN parameter can be specified on the CREATE and UPDATE commands to define a transaction with a Transaction Code Input edit routine. The edit routine must be included in one of the IMS.SDFSRESL concatenated data sets.

The Transaction Code Input edit routine must store the edited message segment to be returned to IMS in the buffer that is addressed by register 1. If the input was processed by the IMS Basic Edit routine, this buffer is always 10 bytes greater than the 2-byte binary count at the beginning of the message segment, and the message segment can be expanded or reduced to any size. The format of the edited message segment in the buffer on return to IMS must be two bytes of binary count, followed by bytes 3 and 4 unchanged from the original message and edited text.

If the input was processed by MFS, the length of this buffer is in the first two bytes of the buffer. No extra space is provided in this buffer for edit routines.

This edit routine is called only when a transaction is entered from a terminal; it is not called when the transaction is inserted by a program-to-program switch or for LU 6.2 terminals.

If specified, a Transaction Code Input edit routine gains control after each message data segment is processed by the IMS Basic Edit routine or MFS, and after transaction code validity and security are checked. If the transaction code is the only data in the message segment and the transaction is a conversational transaction, the edit routine is not entered.

The following table shows the attributes of the Transaction Code (Input) Edit exit routine.

Table 121. Transaction code (input) edit exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.

Table 121. Transaction code (input) edit exit routine attributes (continued)

Attribute	Description
Naming convention	This name must be alphanumeric (A-Z, 0-9, #, $, and @). The name cannot include a blank, comma, period, hyphen, or equal sign, and cannot include the wildcard characters * or %.
Including the routine	The edit routine must reside in one of the IMS.SDFSRESL concatenated data sets.
IMS callable services	To use IMS callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. • Use the ECB found in register 9 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.ADFSSMPL (member name DFSCSMB0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of the buffer location of the input message segment after translation to EBCDIC and after IMS Basic Editing. The first two bytes of the buffer contain the binary message length. The third byte of the buffer is binary zeros. The binary count includes the 4-byte prefix. If Basic Edit is used, the fourth byte of the message segment (Z2) is X'00'. If MFS is used, the fourth byte can contain either a X'01', X'02', or X'03' signifying that option 1, 2, or 3 respectively was selected for the message by the format designer. The fifth byte contains the first byte of the message text. If the input was processed by MFS, the length of this buffer is in the first two bytes of the buffer. No extra space is provided in this buffer for edit routines.
7	CTB address of the physical terminal from which the message is entered.
9	Address of CLB for the communication line from which the message is entered.
10	Address of SMB.
11	Address of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of edit routine. The entry point name and load module name for an edit routine must be the same as the name used for the edit routine in system definition.

Contents of registers on exit

On return to IMS, all registers must be restored except for register 15, which must contain one of the return codes shown in the following table. Register 1 contains the message number if register 15 contains

a value of 12; otherwise it is ignored. Any other value causes the message to be canceled and the terminal operator to be notified.

Return code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled and the message identified by register 1 is sent to the terminal.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“Initialization of IMS callable services (DFSCSII0)” on page 16
Some exit routines must initialize IMS callable services before using them. To initialize IMS callable services, you can issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Sample transaction code (input) edit routine (DFSCSMB0)

Use the sample transaction code (input) edit routine (DFSCSMB0) to define a multisegment transaction named ICS and allow further input flexibility.

Assume a multisegment transaction named ICS. Normally, the first segment of this message contains ICS GN (meaning to get the next segment of a given message), or it contains ICS CAN (meaning to cancel this message). A user-supplied edit routine allows further input flexibility, as shown in the following decision table.

Segment	Message as received and edited by IMS	Message as reedited by user edit routine
First Segment	ICS GN	As received
	ICS	ICS GN
	ICS CAN	Cancel message
	Any other	Cancel message
Other Segment	GN	As received
	CAN	Cancel message
	Any other	Cancel message

The Transaction Code edit routine allows the input for the ICS GN message segment to be shortened.

Chapter 4. IMS system exit routines

Use IMS system exit routines to maintain, enhance, or extend your IMS online system or IMSplex.

Buffer Size Specification facility (DSPBUFFS)

When you use serial access, the Buffer Size Specification facility allows you to control the number of buffers used for RECON data sets when either the local shared resource (LSR) or the nonshared resource (NSR) buffering option is used.

When you use parallel RECON access, VSAM RLS manages a system-wide buffer tool. In this case, you cannot control the number of buffers on a data set basis.

Subsection:

About this facility

DBRC provides a CSECT, DSPBUFFS, for you to override the default number of buffers used. The values in the CSECT are used to build the VSAM local shared resource pool for LSR support or to specify the number of index and data buffers if NSR buffering mode is used.

This facility can be used in DBRC environments.

Binding the CSECT

After assembling the source code, bind the object code of the CSECT into the IMS load module DSPCINT0.

DSPBUFFS layout

The following code sample shows the layout of the DSPBUFFS CSECT. You can assemble your own version of this CSECT and replace it in load module DSPCINT0 using the standard binder setup included in the System Modification Program (SMP) process, or modify the existing version of the CSECT supplied by IBM.

```
DSPBUFFS CSECT ,               DECLARE NBR OF INDEX & DATA BUFFERS
         DC    CL8'DSPBUFFS'   REQUIRED EYECATCHER FOR DUMPS
*
*  DECLARE THE NUMBER OF INDEX AND DATA BUFFERS TO BE USED IN EACH
*  OF THE DEFINED OPERATING MODES WHEN USING THE LSR OPTION OF VSAM.
*  APPLIES TO AN ESA* OR XA ENVIRONMENT ONLY. BOTH BUFFER NUMBERS GIVEN
*  IN EACH CASE MUST BE AT LEAST 4 ELSE DBRC REVERTS TO NSR MODE USING
*  THE NSR BUFFER NUMBERS BELOW THAT CORRESPOND TO THE SAME OPERATING
*  MODE.  THIS FEATURE CAN BE USED TO INHIBIT THE USE OF LSR IN ANY OF
*  THE OPERATING MODES SHOULD SOME PROBLEM ARISE.  REMEMBER THAT UNDER
*  LSR THE INDEX/DATA BUFFERS DEFINED APPLY TO ALL THE ACTIVE RECONS.
*
LSRONLIN DC    AL2(60,120)       IMS ONLINE DBRC
LSRCICS  DC    AL2(60,120)       CICS USE OF DBRC
LSRBATCH DC    AL2(60,120)       OFFLINE/BATCH DBRC
*
*  DECLARE THE NUMBER OF INDEX AND DATA BUFFERS TO BE USED IN EACH
*  OF THE DEFINED OPERATING MODES WHEN USING THE NSR OPTION OF VSAM.
*  APPLIES IF THE LSR OPTION HAS BEEN INHIBITED ABOVE FOR ONE OR
*  MORE OF THE DEFINED OPERATING MODES. THE MINIMUM NUMBER OF INDEX
*  AND DATA BUFFERS ASSIGNED TO EACH RECON IS TWO.
*  REMEMBER THAT UNDER NSR THE NUMBER OF INDEX/DATA BUFFERS
*  DEFINED APPLY TO EACH OF THE RECONS.  NOT SHARED AS WITH LSR.
*
NSRONLIN DC    AL2(2,2)          IMS ONLINE DBRC
NSRCICS  DC    AL2(2,2)          CICS USE OF DBRC
NSRBATCH DC    AL2(2,2)          OFFLINE/BATCH DBRC
         END
```

As the comments and structure of preceding code sample indicate, the first three pairs of halfwords control the number of index and data buffers that are used for LSR. The second three pairs of halfwords

control the number of index and data buffers that are used for NSR. DBRC always uses the VSAM LSR option unless it is inhibited through DSPBUFFS (see comments in the CSECT to see how this is done).

In either LSR or NSR mode, DBRC determines which pair of index/data values to use based on the "operating mode" for each execution. During initialization, DBRC:

1. Uses LSR/NSR pair 1 for IMS control regions
2. Uses LSR/NSR pair 3 for batch jobs or utilities

In effect, by changing or creating your own version of DSPBUFFS, you can specify separate buffering values for batch and online environments. If NSR buffering is used, individual values for BUFNI and BUFND can be specified in the JCL DD statements used to override the default buffer size. For VSAM LSR, only the first three pairs of values are used, so there is no advantage in allocating the RECON data sets through JCL and specifying BUFNI or BUFND values. Similarly, the BUFFERSPACE parameter used when defining a RECON data set through Access Method Services (AMS) is only applicable to the NSR buffering technique and is not used for LSR.

Because the VSAM LSR pools are built while the RECON data sets are open in NSR mode, values for the BUFFERSPACE, BUFNI, and BUFND parameters should not be specified when defining the VSAM clusters and when allocating the RECON data sets using JCL. Because the VSAM LSR pools are built prior to opening the RECON data sets for LSR, supplying values for BUFFERSPACE, BUFNI, or BUFND that exceed VSAM's minimum default only increases the virtual storage needed to support DBRC for batch regions.

Use DSPBUFFS to specify the number of buffers for NSR, even though it is optional. With NSR specified, more efficient use of virtual storage can be achieved than by using the BUFFERSPACE parameter (when defining the RECON clusters) and adjusting the number of index and data buffers through the use of JCL. As a result, the RECON data sets can be dynamically allocated in nearly all applications.

Using IMS callable services with this routine

IMS callable services are not applicable for use with this exit routine.

Example of specifying buffers

Review this example of specifying buffers to see how the Buffer Size Specification facility (DSPBUFFS) overrides the number of buffers to expand the total amount of buffer storage used.

Company XYZ shares RECON data sets between two processors. Processor A is an ESA machine, processor B is not— a coexistence environment involving an earlier release of IMS is on processor B. In this case, each IMS system uses a separate copy of the following example.

XYZ frequently runs batch jobs using DBRC under TSO. However, tight region restrictions exist for jobs run under TSO, so they must limit the amount of storage used by DBRC in these circumstances. However, DBRC storage is not limited when executing as a control region task, so they have replaced DSPBUFFS with the following values:

DSPBUFFS example

```
DSPBUFFS CSECT ,                    DECLARE NBR OF INDEX & DATA BUFFER
         DC    CL8'DSPBUFFS'        REQUIRED EYECATCHER FOR DUMPS
*
*                                   processor A (LSR) SETUP
LSRONLIN DC    AL2(10,26)           ESA ENVIRON - IMS ONLINE DBRC
LSRCICS  DC    AL2(6,12)            ESA ENVIRON - CICS USE OF DBRC
LSRBATCH DC    AL2(6,14)            ESA ENVIRON - OFFLINE/BATCH DBRC
*
*                                   processor B (NSR) SETUP
NSRONLIN DC    AL2(4,9)             NONESA ENVIRON - IMS ONLINE DBRC
NSRCICS  DC    AL2(2,2)             NONESA ENVIRON - CICS USE OF DBRC
NSRBATCH DC    AL2(3,5)             NONESA ENVIRON - OFFLINE/BATCH DBRC
         END
```

When run as an IMS online region, DBRC in processor A (LSR) creates 10 index buffers and 26 data buffers to be shared between the 2 active RECON data sets. In processor B (NSR), DBRC assigns 4 index buffers and 9 data buffers to each RECON data set. When both active RECON data sets are opened for

NSR, a total of 8 index and 18 data buffers are implied. Remember that under NSR, when the spare RECON data set is opened, it too will be assigned 4 index and 9 data buffers. For brief periods of time in processor B, the total number of index and data buffers used are 12 and 27, respectively.

Under LSR, when the spare RECON data set is opened (initially in NSR mode, a VSAM requirement), it is assigned 2 index and 2 data buffers. These values cannot be overridden. For brief periods of time in processor A, the total number of index and data buffers used are 12 and 28, respectively. Thus the total amount of storage that is used for RECON buffers is approximately the same in both processors.

When running batch jobs, DBRC in processor A creates 6 index buffers and 14 data buffers to be shared between the 2 active RECON data sets. In processor B, DBRC assigns 3 index buffers and 5 data buffers to each RECON data set opened with NSR buffering. Again, during those periods of time that all 3 RECON data sets are open, the total amount of buffer storage used is approximately the same in both processors (8 index and 16 data buffers in processor A, 9 index and 15 data buffers in processor B).

Command Authorization exit routine (DFSCCMD0)

The Command Authorization exit routine (DFSCCMD0) can be used to verify that a command is valid from a particular origin. DFSCCMD0 is an optional exit routine for commands entered from IMS terminals, including LU 6.2 and OTMA.

This topic contains Product-sensitive Programming Interface information.

DFSCCMD0 is a required exit routine if it is specified to authorize commands entered from:

- ICMD DL/I calls (from automated operator applications)
- z/OS MCS or E-MCS consoles

This exit routine verifies that the user is authorized to issue a particular command. IMS does not call this exit routine for internally generated or auto-restart commands.

Subsections:

- “About this routine” on page 319
- “Communicating with IMS” on page 321

About this routine

You can use the Command Authorization exit routine with a security product, such as RACF. The return code that the exit routine issues ultimately determines the success or failure of the command authorization; the exit routine can override the outcome of RACF.

The Command Authorization exit routine is optional. For the latest version of DFSCCMD0, see the IMS.SDFSSMPL library; the member name is DFSCCMD0. This sample includes routines for terminals defined using the Extended Terminal Option (ETO) feature, commands entered with ICMD calls, and commands entered from MCS/E-MCS consoles.

Restriction: The Command Authorization exit routine cannot be used to secure type–2 commands; it can secure only type–1 commands. Use the OM user exit routine to secure type-2 commands.

The following table shows the attributes for the Command Authorization exit routine.

Table 122. Command authorization exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	You must name this exit routine DFSCCMD0.
Link editing	You can assemble the sample exit routine or one that you write using the standard IMS macro and copy files. You must manually link edit this routine with DFSCSI00 to use IMS callable services.

Table 122. Command authorization exit routine attributes (continued)

Attribute	Description
Including the routine	DFSCCMD0 must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. If specify one of those parameters and you do not include DFSCCMD0, IMS system initialization ends with a U0718 abend. This routine is required if one or both of the following parameters is specified in the IMS, DBC, or DCC procedures: • AOIS=A or C • CMDMCS=B or C Otherwise, the routine is optional. DFSCCMD0 must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. If specify one of those parameters and you do not include DFSCCMD0, IMS system initialization ends with a U0718 abend. This routine is required if one or more of the following parameters are specified in the IMS, DBC, DCC, or CSL procedures: • AOIS=A or C • CMDMCS=B or C • CMDSEC=A or E Otherwise, the routine is optional.
IMS callable services	This exit routine can use callable storage services. DFSCCMD0 is defined to IMS as a standard user exit. Exit routines that are defined to IMS receive the callable service token in the standard exit parameter list. This exit routine must issue an initialization call (DFSCSII0) to use callable services and you must manually bind with DFSCSI00.
Sample routine location	IMS.ADFSMPL

Using the routine with AO (Automated Operator) applications that issue CMD or ICMD calls

The Command Authorization exit routine can be used with automated operator (AO) applications that issue a CMD or ICMD call. The routine is called for AO applications that issue ICMD calls when the AOIS parameter is specified as A or C in the IMS, DBC, or DCC procedure. The routine is called for AO applications that issue CMD calls when the AOI1 parameter is specified as A or C in the IMS or DCC procedure.

DFSCCMD0 is called during CMD and ICMD processing to check that the AO application is authorized to issue the command that it issued. DFSCCMD0 lets you secure commands issued in the CMD and ICMD calls at the command verb, keyword, and resource name level.

Using the routine with LU 6.2 application programs

When an IMS command is received from an LU 6.2 application program, the Command Authorization exit routine is called. The exit routine is called after a RACF (or equivalent) call is made, regardless of the result of the RACF security check. If neither RACF or the Command Authorization exit routine is available to authorize the command, a default level of command security is provided by IMS for commands from LU 6.2 application programs. The commands included in the default are `/BROADCAST`, `/LOG`, and `/RDISPLAY`.

Using the routine with static terminals

The Command Authorization exit routine can be used with terminals defined statically at system definition. The return code from the default security is passed to the Command Authorization exit routine. IMS calls the exit routine (if it is included in the system) regardless of the result of the default security check; the return code from the exit routine determines authorization.

The DFSCCMD0 exit for terminal security does not require any IMS configuration but just requires that a module of name DFSCCMD0 be available to IMS at startup. This module can be in a STEPLIB library or a link list library. If the DFSCCMD0 module is available in any library that is accessible to Program Fetch during the startup of the IMS control region, then IMS will use it.

Using the routine with ETO terminals

The Command Authorization exit routine can be used with terminals that are defined dynamically using ETO. If RACF (or an equivalent security product) is requested and the user is signed on, RACF performs the command authorization. IMS passes the RACF return code to the Command Authorization exit routine. IMS calls the exit routine (if it is included in the system) regardless of the result of the RACF security check.

If RACF is not requested but the Command Authorization exit routine is included in the system, IMS calls the exit routine and performs command authorization only. If neither RACF nor the Command Authorization exit routine is included, IMS provides command authorization equivalent to the default security available for static terminals.

The `/SIGN` and `/RCLSDST` commands are the only commands that can be entered from an ETO terminal before signon. Although these commands cause IMS to call the Command Authorization exit routine, neither RACF nor the exit routine authorizes the commands.

Using the routine with commands from MCS/E-MCS consoles

This exit routine can be used with commands entered from MCS/E-MCS consoles. The routine is called for commands from MCS/E-MCS consoles when the CMDMCS parameter is specified as B or C in the IMS, DBC, or DCC procedure.

DFSCCMD0 is called during command processing to check that the console is authorized to issue the command. DFSCCMD0 lets you secure commands at the command verb, keyword, and resource levels.

Using the routine with IMS Open Transaction Manager Access

The Command Authorization exit routine can be used with IMS Open Transaction Manager Access (OTMA).

Using the routine in a shared-queues environment

When running in a non-shared-queues environment, the name in the field CNTNAME1 of the CNT representing the WTOR LTERM will be WTOR. In a shared-queues environment, the name in field CTNAME1 of the CNT representing the WTOR LTERM will be IMSID if it is running in a non-XRF environment and RSENAME if it is running in an XRF environment.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “IMS standard user exit parameter list” on page 5 (Version 1)
13	Address of the save area. Your exit routine must not change the first three words of this save area.
14	Return address of IMS.
15	Entry point address of exit routine.

The macro DFSCCMD generates the DSECT for the function-specific parameter list passed to DFSCCMD0 by IMS. For additional information, see DFSCCMD included in IMS.ADFSMAC.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains the return code. See the following table:

Register	Contents	
15	One of the following return codes:	
	Return code	**Meaning**
	0	USER/TERMINAL is authorized to use command
	4	USER/TERMINAL is not authorized
	Negative value	USER/TERMINAL is not authorized. The specified user message is sent to the terminal where command originated.

Related concepts

Defining security during DB/DC and DCCTL system definition (System Administration)

Related reference

"CSL OM user exit routines" on page 589
You can write OM user exits to customize and monitor the OM environment. No sample exits are provided.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

DBRC Command Authorization exit routine (DSPDCAX0)

The DBRC Command Authorization exit routine (DSPDCAX0) can be used to verify that a user is authorized to issue a particular command or DBRC application programming interface (API) request.

Subsections:

About this routine

DSPDCAX0 is an optional exit routine and is selected using the following DBRC commands:

- BACKUP.RECON
- CHANGE
- CLEANUP.RECON
- DELETE
- GENJCL
- INIT
- LIST
- NOTIFY
- RESET.GSG
- REPAIR.RECON

DSPDCAX0 can be used with RACF or another security product. The security product is invoked first, and return and reason codes are passed to DSPDCAX0. The return code from DSPDCAX0 then determines the success or failure of the authorization. DSPDCAX0 overrides the outcome of the security product. DBRC messages issued as a result of unsuccessfully invoking the security product are suppressed.

DSPDCAX0 is required if the COMMAND AUTH setting in the RECON status record is EXIT or BOTH.

DSPDCAX0 must be found in an authorized library or in LINKLST. If DSPDCAX0 is found in a concatenated STEPLIB or JOBLIB, only the data set containing DSPDCAX0 must be authorized. If DSPDCAX0 is found in LINKLST, no authorization check is performed.

The following table shows the attributes for the DBRC Command Authorization exit routine.

Table 123. Command authorization exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	You must name this exit routine DSPDCAX0.
Binding	You must bind this routine into an authorized data set as a separate reentrant (RENT) load module, DSPDCAX0.
Including the routine	No special steps are needed to include this routine. The exit is only included if DBRC command authorization (CMDAUTH) is set to EXIT or BOTH.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	DSPDCAX0 is provided in the IMS.SDFSSMPL data set, and you can modify it to work in both BPE and non-BPE DBRC environments.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routines.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the DBRC command authorization exit parameter list
13	Address of the save area
14	Return address to DBRC
15	Entry point address of exit routine

The following table lists the exit parameter list. It is mapped by the DBRC Command Authorization (DCA) Interface Parameter Block (DSPDCABK).

Table 124. DCA Interface Parameter Block description

Field name	Offset	Length in bytes	Field Usage	Description
DCABLKID	X'00'	X'08'	Input	Eye catcher "DSPCABK"
DCABLKLN	X'08'	X'04'	Input	Length of the block
DCARNPTR	X'0C'	X'04'	Input	Address of the resource name (RN)
DCARNLEN	X'10'	X'04'	None	Resource name length
DCARHPTR	X'14'	X'04'	Input	Address of RN high-level qualifier
DCARHLEN	X'18'	X'04'	Input	Length of RN high-level qualifier
DCARVPTR	X'1C'	X'04'	Input	Address of RN command verb
DCARVLEN	X'20'	X'04'	Input	Length of RN command verb

Table 124. DCA Interface Parameter Block description (continued)

Field name	Offset	Length in bytes	Field Usage	Description
DCARMPTR	X'24'	X'04'	Input	Address of RN command modifier
DCARMLEN	X'28'	X'04'	Input	Length of RN command modifier
DCARQPTR	X'2C'	X'04'	Input	Address of RN command qualifier
DCARQLEN	X'30'	X'04'	Input	Length of RN command qualifier
DCAUserID	X'34'	X'08'	Input	User ID of command issuer
DCAExitAddr	X'3C'	X'04'	None	Address is 0 for BPE user exit
DCAFlags	X'40'	X'04'	Input	Miscellaneous flags: **X'80'** Security product was called. **X'40'** Security exit DSPDCAX0 was called. **X'20'** 1st call (REQUEST=LIST) done. **X'10'** DBRC API Request **X'08'** BPE user exit was called
DCASAFRetCode	X'44'	X'04'	Input	Security product (RACF or equivalent) return code
DCARACFRetCode	X'48'	X'04'	Input	RACF return code
DCARACFRsnCode	X'4C'	X'04'	Input	RACF reason code
DCAExitRetCode	X'50'	X'04'	Output	Security exit return code
DCAUserAreaPtr	X'54'	X'04'	Input	Address is 0 for BPE user exit
DCAUserAreaLen	X'58'	X'04'	Input	Length is 0 for BPE user exit
DCARACRReq	X'5C'	X'08'	Input	RACROUTE request type
DCAVersion	X'64'	X'04'	Input	Parameter list version number (00000001)
	X'68'	X'20'	None	Reserved

Contents of registers on exit

Before returning to DBRC, the exit routine must restore all registers except for register 15, which contains the following return code.

The following table reflects the register contents for non-BPE based DBRC exit routines.

Register	Contents	
15	One of the following return codes:	
	Return code	**Meaning**
	0	USER is authorized to use the DBRC command.
	nonzero	USER is not authorized to use the DBRC command.

Related reference

"BPE-based DBRC user exit routines" on page 529
The BPE-based DBRC user exit routines enable you to run the existing DBRC user exit routines in a BPE (Base Primitive Environment).

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"DBRC Security exit routine" on page 531
The DBRC Security exit routine can be used to verify that a user is authorized to issue a particular command or DBRC application programming interface (API) request.

DBRC SCI registration exit routine (DSPSCIX0)

The DBRC SCI Registration exit routine (DSPSCIX0), formerly called the SCI Registration exit routine, supplies the IMSplex name needed for DBRC's Structured Call Interface (SCI) registration.

Subsections:

About this routine

The DBRC SCI Registration exit routine (DSPSCIX0) is called by DBRC before registering with the SCI. DSPSCIX0 supplies the IMSplex name needed for SCI registration. The exit can also supply a DBRC group ID to identify unique RECON sharing groups. If the exit is not used, DBRC will behave as if the sample version of the exit was being used.

DSPSCIX0 must be found in an authorized library or in LINKLST. If DSPSCIX0 is found in a concatenated STEPLIB or JOBLIB, only the data set containing DSPSCIX0 must be authorized. If DSPSCIX0 is found in LINKLST, no authorization check is performed. In a TSO environment, the library must be located in the task library (TASKLIB).

The following table shows the attributes for the DBRC SCI Registration exit routine.

Table 125. DBRC SCI registration exit routine (DSPSCIX0)

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL.
Naming convention	You must name this exit routine DSPSCIX0.
Binding	You must bind this routine into an authorized data set as a separate reentrant (RENT) load module, DSPSCIX0.
Including the routine	No special steps are needed to include this routine. If the exit is not used, DBRC will behave as if the sample version of the exit was being used.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	IMS.ADFSSMPL (member name DSPSCIX0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routines.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the DBRC SCI registration exit parameter list
13	Address of the save area
14	Return address to DBRC
15	Entry point address of exit routine

DSPSCIX0 uses standard linkage conventions. It is passed six parameters using a standard format parameter list. The following six parameters are passed to DSPSCIX0:

- A RECON data set name. Any one of the RECON data set names in use can be passed to the exit.
- A 5 byte area in which DSPSCIX0 returns an IMSplex name.
- If the IMSPLEX EXEC statement parameter is specified, the value of the IMSPLEX parameter is passed as this parameter. The EXEC statement referred to can be on any job step that uses DBRC.
- A full word containing the version of the parameter list.
- A 3-byte area in which DSPSCIX0 returns a DBRC group ID initialized to '001'.
- If the DBRCGRP EXEC statement parameter is specified, the value of the DBRCGRP parameter is passed as this parameter. The EXEC statement referred to can be on any jobstep that uses DBRC.

Contents of registers on exit

Before returning to DBRC, the exit routine must restore all registers except for register 15, which contains the following return code.

Register	Contents	
15	One of the following return codes:	
	Return code	**Meaning**
	0	DBRC expects a valid IMSplex name and DBRC group ID to be returned in the parameter list. The IMSplex name and group ID are used for registration with SCI.
	4	Access is attempted without SCI registration. If the RECON indicates that RECON Loss Notification is active or PRA is active, DSP1136A is issued and RECON access fails.
	8	Access is attempted without SCI registration. RECON access is forced without regard to RECON content. DSP1143I is issued. Access to the RECON will be done in serial mode regardless of the access setting in the RECON data sets. If another instance has the RECONs opened in parallel mode, this access will fail with an OPEN failure.
	12	RECON access fails and message DSP1139I is issued.
	Any other value	Will behave as RC12 in this implementation.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

“IMS standard user exit parameter list” on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

“BPE-based DBRC user exit routines” on page 529

The BPE-based DBRC user exit routines enable you to run the existing DBRC user exit routines in a BPE (Base Primitive Environment).

Sample DBRC SCI registration exit routine

Use the sample DBRC SCI registration exit routine to return the IMSPLEX parameter value and the group ID value specified by the DBRCGRP EXEC parameter.

For the latest version of DSPSCIX0, see the IMS.ADFSSMPL library, member name DSPSCIX0.

The sample version of DSPSCIX0 will issue a return code 4 in register 15 unless an IMSplex name is supplied through the IMSPLEX EXEC parameter. If an IMSplex name is supplied, DSPSCIX0 will return the IMSPLEX parameter value and the group ID value specified by the DBRCGRP EXEC parameter. If an IMSplex EXEC parameter is specified but no DBRCGRP EXEC parameter is specified, the sample exit will return the IMSplex parameter value and the default group ID '001'.

The sample version of DSPSCIX0 contains a table of RECON data set names and associated IMSplex names and DBRC group IDs. As shipped, the exit responds to any RECON name with return code 4 and the table has no other entries. To activate the RECON Loss Notification or use parallel RECON access, either specify an IMSplex name through the IMSPLEX EXEC parameter on all jobs which use DBRC, or add RECON data set names, associated IMSplex names, and DBRC group IDs to the table.

The first entry in the table follows the label PLEXTABL. Each entry consists of a 44-byte RECON data set name, left justified and padded with blanks, followed by a 5-byte character IMSplex name, a 3-byte group ID, and a 4-byte hexadecimal return code. The last entry is the default entry consisting of an asterisk (*) for a RECON data set name and, unless altered by the user, a blank IMSplex name, a default group ID '001,' and a return code of 4. While the default response can be changed, the entry containing the asterisk marks at the end of the table should not be removed unless the associated exit logic is changed as well.

A table modified for a production IMSplex and a test IMSplex could appear as follows:

```
PLEXTABL DS   0H
* production RECONs and associated IMSplex
         DC   CL44'PROD.RECON1'    RECON name
         DC   CL5'PLEXA'            IMSplex name
         DC   CL3'GP1'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
         DC   CL44'PROD.RECON2'    RECON name
         DC   CL5'PLEXA'            IMSplex name
         DC   CL3'GP1'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
         DC   CL44'PROD.RECON3'    RECON name
         DC   CL5'PLEXA'            IMSplex name
         DC   CL3'GP1'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
* test RECONs and associated IMSplex
         DC   CL44'TEST.RECON1'    RECON name
         DC   CL5'PLEXT'            IMSplex name
         DC   CL3'GT1'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
         DC   CL44'TEST.RECON2'    RECON name
         DC   CL5'PLEXT'            IMSplex name
         DC   CL3'GT1'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
         DC   CL44'TEST.RECON3'    RECON name
         DC   CL3'GT1'              Group ID
         DC   CL5'PLEXT'            IMSplex name
         DC   XL4'00000000'         RC00 = use the IMSplex name
* 2nd test RECON group - same plex as Test RECON group 1
         DC   CL44'TEST2.RECON1'    RECON name
         DC   CL5'PLEXT'            IMSplex name
         DC   CL3'GT2'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
         DC   CL44'TEST2.RECON2'    RECON name
         DC   CL5'PLEXT'            IMSplex name
         DC   CL3'GT2'              Group ID
         DC   XL4'00000000'         RC00 = use the IMSplex name
         DC   CL44'TEST2.RECON3'    RECON name
         DC   CL3'GT2'              Group ID
         DC   CL5'PLEXT'            IMSplex name
         DC   XL4'00000000'         RC00 = use the IMSplex name
* end of table - default exit response is not to use SCI for unknown RECONs
         DC   CL44'*'              RECON name
```

```
         DC   CL5'     '            unusable IMSplex name
         DC   CL3'001'              Default Group ID
         DC   Xl4'00000004'         RC04 = no SCI registration
```

Dependent Region Preinitialization routines

Dependent Region Preinitialization routines enable you to perform any application-unique dependent region initialization.

Subsections:

About these routines

Dependent Region Preinitialization routines can activate any z/OS system or data management services for which they are authorized, although they cannot issue DL/I calls or activate IMS system services. Because they receive control after module preload, but before IMS scheduling, you might want to use these routines for such tasks as building an internal table for your applications to access during dependent region processing.

For example, you can use a preinitialization routine to build a table for application decision making. You can maintain this table by using z/OS services in the following manner:

- Using z/OS storage management services, the preinitialization routine can acquire and format a main storage table.
- Using z/OS Name/Token callable services, the preinitialization routine can establish a name/token pair for the storage that provides the user application access to the storage area.

This name/token pair can then be used by the dependent region applications using the Name/Token services to access the table. It is your responsibility to determine what these preinitialization routines do, and how the information is made available to user applications.

Preinitialization routines are not intended to control the IMS dependent region environment. These routines provide installation information that can be shared between applications. This information can be used to control the applications and allow the application to make decisions based on the information in these tables.

The preinitialization routines must not be system-type routines (for example, z/OS services, Language, or Access Method) but rather user-written routines.

Related Reading: For guidance-level information to help you decide whether or not you want to write these routines to initialize dependent regions, see "Establishing IMS Security" in *IMS Version 15.2 System Administration*.

The following table shows the attributes of Dependent Region Preinitialization routines.

Table 126. Dependent region preinitialization routines attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	Using standard z/OS conventions, you can give the routine any name up to eight characters in length. Be sure that the name is unique and does not conflict with the existing members of the data set in which this routine is stored. Because most IMS-supplied routines begin with the prefix "BPE," "CQS," "CSL," "DFS," "DBF," "DSP," or "DXR," choose a name that does not begin with these letters.

Table 126. Dependent region preinitialization routines attributes (continued)

Attribute	Description
Binding	Before a dependent region can be initialized, you must assemble all required dependent region preinitialization routines and bind into a concatenation of // STEPLIB. Normally, this is IMS.PGMLIB or the associated application program library. You must bind the routine as *reentrant* (RENT).
Including the routine	See this topic.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	None.

Activating the routine

The Dependent Region Preinitialization routines get control after the dependent region has IDENTIFIED or SIGNED-ON to the associated Control Region, but before IMS scheduling is attempted. These routines execute under the IMS Program Control Task whenever it:

- Is attached or reattached in problem program state/user key 8
- Receives control in the order specified in the PROCLIB member

Each Preinitialization exit routine is identified by an 80-byte record in a DFSINT*xx* member of IMS.PROCLIB, where *xx* is a suffix specified by the PREINIT keyword of the IMS dependent region procedures IMSBATCH, DFSMPR, and IMSFP.

Related Reading: For details about these procedures, see *IMS Version 15.2 System Definition*.

Each record identifies one program in IMS.PGMLIB that is to receive control during dependent region initialization (or reinitialization after an IMS user application program abnormal termination).

The 80-byte record identifying each preinitialization routine is as follows:

Column	Contents
1-71	Routine names and entry points. The last name on a record is denoted by a comma followed by one or more blanks; the last name on the last record is followed by one or more blanks.
72-80	Must remain blank. Ignored.

The routines are given control in the order specified in the member. If a requested routine is not found, the dependent region abnormally terminates with a U0588.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routines.

Contents of registers on entry

On entry, the routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Zero.
13	Address of save area. The routine must not change the first three words.
14	Return address to IMS.

Register	Contents
15	Entry point of routine.

Contents of registers on exit

Before returning to IMS, the routine must restore all registers except register 15, which must contain the following:

Register	Contents
15	0

Dump Override Table (DFSFDOT0)

Use the Dump Override Table to either force or suppress dumps for specified abends.

- "About this table" on page 330

About this table

The IMS Dump Override Table is used to override default dump processing for IMS abends that occur after early IMS initialization. You can use this table to force a dump to be taken for abend codes for which dumps are normally suppressed. You can also use it to prevent dumps for abend codes for which dumps are normally taken.

If DFSFDOT0 is present in IMS.SDFSRESL, IMS will load it during IMS initialization. If an abend occurs, this table will be searched for an entry that matches the abend code. If a matching entry is found, IMS will either create a memory dump or not create a memory dump based on the action specified on the entry's DFSFDOT macro invocation (FORCE or SUPPRESS). If a matching entry is not found, or if DFSFDOT0 is not present in IMS.SDFSRESL, IMS will use its default logic to decide whether or not to create a memory dump.

The Dump Override Table suppresses only IMS Control Region, IMS DLS Region, and DBRC Region abend dumps. IMS Dependent region dumps cannot be suppressed with the Dump Override Table.

The only change that the Dump Override Table makes to the dumping process is to force or suppress the initial dump decision. IMS still creates only one dump, even when multiple abends occur and matching entries are found in the Dump Override Table.

A sample Dump Override Table is shipped with a default set of entries. The entries in this sample are the same as the default processing that IMS performs if there is no DFSFDOT0 present in IMS.SDFSRESL. Modify DFSFDOT0 to fit your own needs. If you want no entries in the Dump Override Table, you must create a DFSFDOT0 with no entries. After you assemble your customized version, link it into the system to activate the changes.

The following table shows the attributes of Dump Override table.

Table 127. Dump Override Table attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL.
Naming convention	This table must be named DFSFDOT0.
Link editing	This table has no executable code. It must be linked into an authorized data set as a separate, serially reusable (REUS) load module, DFSFDOT0.
Including the routine	No special steps are needed to include this table. If DFSFDOT0 does not exist, IMS will use the default dump override values that are included in DFSFDMP0.
IMS callable services	IMS callable services are not applicable for use with this table.

Table 127. Dump Override Table attributes (continued)

Attribute	Description
Sample routine location	IMS.ADFSSMPL (member name DFSFDOT0).

DFSFDOT macro

The DFSFDOT macro is an IMS-provided macro that is used to generate the DFSFDOT0 table. A DFSFDOT macro call must be coded for each abend for which you want to force a dump to be taken, and for which you want to suppress a dump. Though this macro is used to build the Dump Override Table, the macro is separate from the Dump Override Table to maintain IMS integrity.

DFSFDOT parameters and descriptions

Parameters are required and must be specified when defining the Dump Override Table.

DFSFDOT BEGIN
This parameter is required at the start of the Dump Override Table definition. It must be coded before any other DFSFDOT invocations. When BEGIN is specified, no other options are allowed. If any options are specified, they are ignored.

DFSFDOT END
This parameter is required at the end of the Dump Override Table and must be the last DFSFDOT invocation in DFSFDOT0. When END is specified, no other options are allowed. If any options are specified, they are ignored.

ABEND=
This parameter specifies a user or system abend for which a dump is either to be forced or suppressed. The abend is specified in one of the following forms:

UNNNN, where NNNN is the four-digit decimal number (U0780, U4095) of the abend.
SXXX, where XXX is the three-digit hexadecimal number (S075, S3E7) of the abend.

DUMP=
This parameter specifies whether the abend dump is forced or suppressed. This parameter overrides IMS dump decision logic and the z/OS dump request bit. It has two options:

FORCE generates a dump for a non-dumping ABEND. There is no default value for DUMP=.
SUPPRESS prevents unwanted dumps. Default = none.

The Dump Override Table can specify an abend code and an action of SUPPRESS. However, IMS cannot suppress all dumps. For example, z/OS or another component can write the dump prior to IMS receiving control. In the case of system abend code S122, z/OS causes the dump to be written before the abend is issued and before IMS receives control. IMS then issues message DFS3984I stating that the dump has been suppressed. This message is misleading, but as far as IMS is concerned the dump has been suppressed. IMS cannot suppress dumps produced by abends that occur after IMS has already processed the Dump Override Table. In the case of ABENDU0002, IMS has already processed the Dump Override Table.

IMS documentation does not explicitly list every abend that supplies a dump that cannot be suppressed by using the Dump Override Table.

Sample Dump Override Table (DFSFDOT0)

This example shows you how to use the DFSFDOT to either force or suppress dumps for specified abends using a Dump Override Table.

The table in this example forces dumps for ABENDS S075, U780, and S222; the table suppresses dumps for ABENDS S80A and U790.

```
DFSFDOT BEGIN
DFSFDOT ABEND=S075,DUMP=FORCE
DFSFDOT ABEND=U0780,DUMP=FORCE
DFSFDOT ABEND=S80A,DUMP=SUPPRESS
DFSFDOT ABEND=S222,DUMP=FORCE
```

```
DFSFDOT ABEND=U0790,DUMP=SUPPRESS
DFSFDOT END
```

Entries need not be in order.

You can generate a Dump Override Table with no FORCE or SUPPRESS by coding a single DFSFDOT BEGIN/END pair, as follows:

```
DFSFDOT BEGIN
DFSFDOT END
```

Errors

Possible errors include:

ASSEMBLY ERROR
An assembly error is issued when an invalid abend code is specified or conflicting dispositions for an abend code are found.

ABEND U0718
An ABEND U0718 (MODULE LOAD FAILURE) is issued if DFSFDOT0 cannot be loaded.

Messages

DFSFDMP0 issues message DFS3984I when a TCB ABEND code matches an entry in the Dump Override Table. The message appears as:

```
DFS3984I DUMP FOR ABEND _____ FORCED BY DUMP OVERRIDE TABLE
DFS3984I DUMP FOR ABEND _____ SUPPRESSED BY DUMP OVERRIDE TABLE.
```

ESAF In-Doubt Notification exit routine (DFSFIDN0)

Use the optional External Subsystem Attach Facility (ESAF) In-Doubt Notification exit routine (DFSFIDN0) to identify external subsystem in-doubt units if an IMS failure occurs.

With this information, you can resolve the in-doubt work before restarting the failed IMS. This routine is optional. If it is not used, IMS attempts to resolve the in-doubt data when it can.

Subsections:

- "About this routine" on page 332
- "Communicating with IMS" on page 333
- "Sample exit routine" on page 334

About this routine

During an emergency restart or an FDBR start, when scanning the units of work for recovery, IMS provides to the user exit routine the identities of all external subsystem units of work, the names of the external subsystems, and the final resolutions of the data.

IMS synchronously calls the exit routine one time for each in-doubt external subsystem unit of work. Because these are synchronous calls, consider the performance impact on FDBR when writing the exit routine.

In an XRF environment, consider the performance impact of the exit routine during an XRF takeover.

Attributes of the ESAF In-Doubt Notification exit routine are described in the following table.

Table 128. ESAF In-Doubt Notification exit routine attributes

Attribute	Description
IMS environments	DB/DC, DCCTL, and DBCTL.

Table 128. ESAF In-Doubt Notification exit routine attributes (continued)

Attribute	Description
Naming convention	The exit routine must be named DFSFIDN0.
Including the routine	To use this optional exit routine, you must name it DFSFIDN0 and link-edit it into an APF-authorized library. This library can be either the JOBLIB or STEPLIB for the FDBR region. To use this exit routine to resolve in-doubt work during an FDBR recovery, you must link-edit it into the IMSVS.RESLIB concatenation of the FDBR procedure.
IMS callable services	This exit routine is not eligible to use IMS callable services. The routine is called in TCB mode with AMODE=31. SVCs are allowed.
Sample routine location	A sample exit named DFSFIDN0 is provided in the IMS.ADFSSMPL library. You must compile the sample exit routine before you can use it.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

The registers on entry contain the following information:

Register	Contents
1	Address of the DFSRNID parameter list.
13	Address of the save area. This save area is not chained to any IMS save area.
14	Return address to IMS.
15	Entry address of this exit routine.

The DFSRNID parameter list contains the following information:

Table 129. ESAF In-Doubt Notification exit routine parameter list

Offset	Length	Field name	Description
0	4	RNIDID	Eye catcher
4	4	RNIDLEN	Length of DFSRNID block
8	2	RNIDVER	Version of DFSRNID
10	2	RNIDREL	Release of DFSRNID
12	4	RNIDIMS	IMS ID This field is included for compatibility with previous releases of IMS. See field RNIDIMSN.
16	4	RNIDSSYS	External subsystem ID that owns in-doubt data

Table 129. ESAF In-Doubt Notification exit routine parameter list (continued)

Offset	Length	Field name	Description
20	2	RNIDRESO	Unit-of-work resolution action: **CO** Commit **AB** Abort
22	2	RNIDUOWL	Unit-of-work length
24	4	Reserved	
28	4	RNIDUOW	Unit-of-work identifier address
32	8	RNIDSST	External subsystem type
40	8	RNIDIMSN	IMS name (UOR owner)

Contents of registers on exit

All registers must be restored on return.

Sample exit routine

Source code for a sample DFSFIDN0 exit routine is provided with IMS in the IMS.ADFSSMPL library. The sample exit routine does not perform any processing on incoming in-doubt UOWs. It is intended to demonstrate the basic program flow that is required for a user-supplied DFSFIDN0 exit routine. The sample performs the following basic processing steps:

1. Receive the RNID.
2. Create a work area, or issue a DFS3723E message if it is unable to do so.
3. Build a DFS3722I message that reports the status of the UOW in the log.
4. Issue the DFS3722I message.
5. Free the work area.
6. Return control to IMS.

Related concepts

Accessing external subsystem data (System Definition)

External Subsystem Attach Facility (ESAF) (Communications and Connections)

Related reference

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

ESAF subsystem exit routines

IMS uses the External Subsystem Attach Facility (ESAF) to activate external subsystem-supplied exit routines. These routines perform prescribed subsystem unique attachment functions.

IMS uses the module names in the external subsystem module table (ESMT) specified for the control region to load the exit routines during control region initialization. The ESMT specified (or defaulted to) for a dependent region is used to load the exit routines into the dependent region.

Most of the exit routines execute functions that are required for attach processing; others are optional. When an exit routine required to support connection processing is not present, IMS terminates the connection to the external subsystem, if one exists, and issues an informational message (DFS3068I). If an application program is involved, it is terminated with a user abend (U3049).

General exit routine interface

This topic describes general interfaces for all the External Subsystem exit routines. You need to familiarize yourself with these interfaces.

Exit parameter list (EPL)

IMS activates an external subsystem exit routine, passing the address of an exit parameter list (EPL) in register 1 (see the following figure). The EPL contains the addresses of the parameters required by the exit routine. IMS passes to an exit routine only the specific parameters it requires, so the contents and length of the EPL differ between exit routines. The parameters for each exit routine are specified in the individual exit routine description topics that follow.

The general format of the EPL is an array of fullword fields (4-byte fields, fullword aligned), each containing the address of a parameter required for the exit routine being activated. The first word in the EPL always contains the address of a 4-byte parameter count field. The binary value in the count field is the number of parameters being passed minus the count parameter (see the following figure).

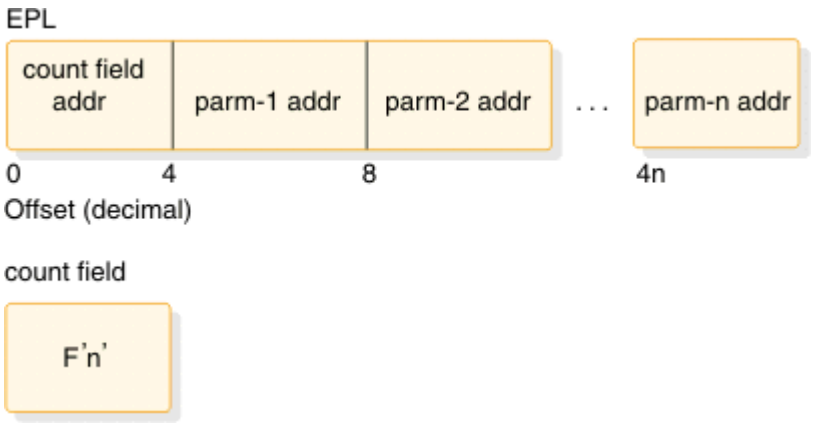


Figure 22. Exit parameter list

Contents of registers

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of exit parameter list (EPL).
13	Address of save area. The exit routine must not change the backward chain field, but it can alter the forward chain field.
14	Return address to IMS.
15	Entry point of exit routine.

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain a return code. IMS provides one standard register save area (address in register 13) in the appropriate storage protect key into which the exit routine can save the entry register contents. The save area backward chain field must not be altered (such as to chain the save area into a save area set). The exit routine can alter the forward chain field.

Return codes

Return codes are exit routine specific. The return codes are shown in hexadecimal format. Return code 20 is supported for all exit routines and is described as follows.

Unsupported return codes

If register 15 on return from an exit routine contains a return code that is not supported for the exit routine, it is treated as an error. IMS terminates the connection for the region that activated the exit routine if one exists. If an application program is involved, it terminates with a U3049 abend.

Return code 20

Return code 20 is used by all exit routines to indicate a 'should not occur' condition and is described as follows:

Return code	Meaning
20	Should not occur. The exit routine encountered a 'should not occur' condition while processing the request. Such conditions include invalid save areas, protocol violations, invalid work areas, and invalid parameter lists. Action: • If an application program is involved, it terminates with abend U3044. If the external subsystem does not respond or responds incorrectly to the control region echo request, the connection to that subsystem terminates. • If the external subsystem does respond, the identify for the dependent region terminates. A subsequent external subsystem request causes the structure to be rebuilt. • If a connection exists when the error is encountered, it terminates by activating the Terminate Identify exit routine.

Related reference

“Abort Continue exit routine” on page 339
The Abort Continue exit routine is activated when the application issues an IMS DL/I ROLB call or an external subsystem votes 'no' to a commit prepare request.

Exit routine interface control blocks

Exit routine interface control blocks can contain the prefix for the external entry vector table and the vector table itself, which contains the addresses of external subsystem exit routine modules.

This topic describes the prefix for the external entry vector table and the vector table itself.

External entry vector table prefix (EEVTP)

The address of an external entry vector table prefix (EEVTP) control block is always passed in the EPL on exit routine activations. The EEVTP is the primary external subsystem interface control block and contains the:

- Address of the external entry vector table (EEVT)
- Address of the resource translation table (RTT)
- Environment indicator (control or dependent region TCB)
- Address of the IMS service exit routine router module

External entry vector table (EEVT)

The external entry vector table (EEVT) contains the addresses of external subsystem exit routine modules. IMS gets exit routine addresses from this control block to activate the exit routines. IMS creates an EEVT (and EEVTP) in the control region and in each dependent region before loading the modules defined in the ESMT into the region. When the modules are loaded their addresses are stored in the EEVT.

The EEVT is an IMS control block, however, module addresses are placed in the control block based on the module definitions contained in the ESMT. Therefore, the external subsystem, in creating the ESMT,

must make sure that exit routine module definitions provide for placement of exit routine addresses in the EEVT according to the EEVT mapping layout used by IMS. The ESAP can manipulate addresses in this vector table if it chooses.

In addition to exit routine modules, the external subsystem can define other modules in the ESMT, for example, modules that would be activated by exit routines and not by IMS. IMS loads all modules defined in the ESMT and stores their addresses as specified in the definitions.

Because of how definition and loading of external subsystem modules is done, it is possible for the external subsystem to 'extend' the EEVT to include the addresses of non-exit-routine modules.

Note: Extending the EEVT to include the addresses of non-exit-routine modules is not recommended. IMS might add fields to the EEVT at a later time, in which case, the external subsystem might have to respecify module definitions (that is, regenerate the ESMT) and recompile modules.

Related concepts

Loading external subsystem modules (Communications and Connections)

Control block mapping

The DFSEEVTP DSECT maps the EEVTP control block and the DFSEEVT DSECT maps the EEVT control block.

The EEVTP is the prefix of the EEVT and contains the address of the EEVT.

DFSEEVTP

The following fields are of interest to the external subsystem:

Offset	Field length	Field name	Description
X'0'	X'4'	EEVPNAME	Eye catcher - 'EEVP'
X'8'	X'4'	EEVPEEA	EEVT ADDRESS
X'10'	X'4'	EEVPEWA	Available for external subsystem
X'14'	X'4'	EEVPRTA	Recovery token address
X'1C'	X'4'	EEVPRTTA	Resource translation table address
X'20'	X'4'	EEVTLDIR	Available for external subsystem
X'28'	X'4'	EEVPESGL	DFSESGL0 address
X'2E'	X'1'	EEVPF1	Environment indicators
-	-	EEVPCR	= X'01'; Control region
-	-	EEVPMPP	= X'02'; MPP dependent region
-	-	EEVPBMP	= X'04'; BMP dependent region
-	-	EEVPIFPN	= X'08'; Fast Path non-message driven
-	-	EEVPIFPM	= X'10'; Fast Path message driven
-	-	EEVPBMPN	= X'20'; Non-message driven BMP
-	-	EEVPBDB2	= X'80'; Batch DB2 region
X'2F'	X'1'	EEVPF2	Environment indicators
-	-	EEVPDRPG	= X'01'; Running under dependent region; program controller TCB

Offset	Field length	Field name	Description
X'34'	X'8'	EEVPSOTN	Signon token
X'3C'	X'4'	EEVPESMT	ESMT address
X'40'	X'4'	EEVPSVA	EESV address
-	-	EEVPLGTH	= X'44'; Length of EEVP

DFSEEVT

The following fields are of interest to the external subsystem:

Offset	Field length	Field name	Description
X'0'	X'4'	EEVTNAME	Eye catcher - 'EEVT'
X'4'	X'4'	EEVTINIT	Initialization exit address
X'8'	X'4'	EEVTID	Identify exit address
X'C'	X'4'	EEVTRID	Resolve indoubt exit address
X'10'	X'4'	EEVTSO	Signon exit address
X'14'	X'4'	EEVTCT	Create thread exit address
X'18'	X'4'	EEVTCP	Commit prepare exit address
X'1C'	X'4'	EEVTCC	Commit continue exit address
X'20'	X'4'	EEVTA	Abort exit address
X'24'	X'4'	EEVTTT	Terminate thread exit address
X'28'	X'4'	EEVTSF	Signoff exit address
X'2C'	X'4'	EEVTTI	Terminate identify exit address
X'30'	X'4'	EEVTSNO	Subsystem not operational exit address
X'34'	X'4'	EEVTSST	Subsystem termination exit address
X'38'	X'4'	EEVTNC	Normal call exit address
X'3C'	X'4'	EEVTECHO	Echo exit address
X'40'	X'4'	EEVTCMD	Command exit address
X'44'	X'4'	EEVTCV	Commit verify exit address
X'48'	X'4'	EEVTIC	Not used
X'4C'	X'4'	EEVTABE	Not used
X'50'	X'4'	EEVTAT	Associate Thread exit address
X'54'	X'4'		Reserved
-	-	EEVTLGTH	= X'58'; Length of EEVT

Abort Continue exit routine

The Abort Continue exit routine is activated when the application issues an IMS DL/I ROLB call or an external subsystem votes 'no' to a commit prepare request.

The Abort Continue exit routine is activated by IMS for all transaction types. The external subsystem resource managers hold onto the resources they have acquired on behalf of the application. The application will continue using the current recovery token.

Subsections:

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset hexadecimal	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'2'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Abort Continue successful. **Action:** IMS continues normal processing.
4	Abort Continue unsuccessful. ESAP or external subsystem processing of the request failed. **Action:** IMS terminates the application with abend U3045 (the input message is discarded; DL/I resources are backed out). The control region performs resolve indoubt processing for the recovery token. The dependent region is terminated; which implicitly terminates the dependent region connection to the external subsystem; the Signoff and Terminate Identify exit routines are not called). BMP jobs must be resubmitted; they resume processing at the prior commit point.
20	Should not occur.

Related reference

"Resolve Indoubt exit routine" on page 358

The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

Associate Thread exit routine

The Associate Thread exit routine is optional and, if provided, is activated only in the dependent region.

If the Associate Thread exit routine is not provided, its function is precluded from dependent region processing and no indication is provided.

Associate Thread exit routine processing is related to the processing of the dependent region Signon exit routine processing. The Signon exit routine is activated for each message processed. Prior to activating the Signon exit routine, the Associate Thread exit routine is activated.

The Associate Thread exit routine, in conjunction with the Create Thread exit routine and the Terminate Thread exit routine, can be used to control the allocation and de-allocation of thread structures.

A thread structure is allocated during the processing of the Create Thread exit routine when an application program is scheduled into a dependent region and makes a request to an external subsystem. When the application program terminates normally, the thread structure is de-allocated during the processing of the Terminate Thread exit routine.

If multiple application programs are repeatedly scheduled in the same dependent region, the associated threads must be allocated and de-allocated for each scheduling. This requires the activation of the Create and Terminate Thread exits for each application scheduling.

The associate Thread exit may be used to allocate a pool of skeletal, inactive thread structures which Create Thread processing will select and activate based upon the current application user ID. Terminate Thread processing will deactivate the thread structure but not de-allocate it. This allows the deactivated structure to be reused when necessary.

This concept will allow for more efficient processing and shorter path length during Create Thread and Terminate Thread exit processing.

The pool of skeletal thread structures may be comprised of multiple structures that have been allocated as a single entity or allocated dynamically as needed. In either case, the possibility exists that additional structures will be needed but cannot be allocated. In this case, a concept of reusing inactive structures on a least recently used algorithm can be implemented.

If associate Thread exit processing is used to pool application's thread structures, the external subsystem must ensure the allocated structures are de-allocated when the target dependent region terminates. IMS does not communicate dependent region terminations to the external subsystem and expects the subsystem to monitor the dependent region TCB with a z/OS End of Task (EOT) exit routine.

The subsystem should then de-allocate the thread structures for the terminating TCB.

Subsections:

- “Activating the routine” on page 340
- “Contents of register 15 on return” on page 341

Activating the routine

The exit routine is activated in key seven supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'3'.

Offset (hexadecimal)	Decimal	Content
4	4	Address of the EEVT prefix.
8	8	Address of the 8-character application program name, left-aligned and padded to the right with blanks. If the application is CPI-C driven, the PSB name that is associated with the APSB call is used instead of the application program name.
C	12	Address of the 8-character PSB name, left-aligned and padded to the right with blanks. If a PSB is not allocated, the name of the SMB is used.

Contents of register 15 on return

Processing of return codes that are received from the exit routine:

Return code	Meaning
0	Associate Thread call successful. **Action:** IMS continues normal processing.
4	Associate Thread call unsuccessful. The external subsystem rejected the specified request. **Action:** IMS activates the Subsystem Not Operational exit routine. The return code from the Subsystem Not Operational exit routine determines further processing.
8	Associate Thread call unsuccessful. The external subsystem was unable to complete the request due to the unavailability of a required resource (resource allocation failure). **Action:** IMS terminates the application program with abend U3048.
C	Associate Thread call unsuccessful. The request failed in the external subsystem. **Action:** IMS terminates the application program with abend U3045.
10	Associate Thread call unsuccessful. The request failed in the external subsystem because communications with it are broken. **Action:** IMS terminates the application program with abend U3044.
14	Associate Thread call unsuccessful. The external subsystem was unable to complete the request due to a definitional conflict. **Action:** IMS terminates the application program with abend U3047.
18	Unsupported.
1C	Unsupported.
20	Should not occur.
24	Associate Thread call unsuccessful. A resource deadlock was detected by the external subsystem. **Action:** IMS terminates the application program with abend U777. All changes are backed out and the application is rescheduled.

Related reference
Signon exit routine (Exit Routines)
Subsystem Not Operational exit routine (Exit Routines)
Terminate Thread exit routine (Exit Routines)
Create Thread exit routine (Exit Routines)

Command exit routine

The Command exit routine allows external subsystem commands to be entered from IMS terminals and Automated Operator Interface (AOI) applications.

IMS activates the optional external subsystem Command exit routine when IMS discovers the subsystem's unique command recognition character (CRC) as the first non-blank character in the text portion of the /SSR command.

IMS passes the command output destination name (LTERM name) to the exit routine. The external subsystem can send a command response to this destination by using the IMS Message Service.

For commands from an AOI program or from an input-only device not associated with an output device, the output destination is the IMS MTO; otherwise it is the inputting terminal.

IMS also provides the user ID associated with the command, if any, that the external subsystem might use for security authorization checking.

IMS sends message DFS3612I to the inputting terminal if an /SSR command is entered and a Command exit routine was not provided by the external subsystem.

Subsections:

- "Activating the routine" on page 342
- "Contents of register 15 on return" on page 343

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB). The following table explains the contents of the EPL.

Table 130. EPL contents

Offset	Offset (decimal)	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'5'.
X'4'	4	Address of EEVT prefix.
X'8'	8	Address of the variable length external subsystem command input. See the next table for the format of the command input.
X'C'	12	Address of the 8-byte alphanumeric destination name (that is, LTERM name) where the command response message, if any, is to be sent. The name is left justified and padded with blanks on the right.
X'10'	16	Address of the 8-character user ID associated with the command input message. The user ID is left justified and padded with blanks on the right. If IMS extended security (SIGNON|SIGNOFF) is not active, or the inputting terminal did not sign on, the user ID field contains the output destination LTERM name.

Table 130. EPL contents (continued)

Offset	Offset (decimal)	Content
X'14'	20	Address of the 8-byte RACF group name for the user ID that entered the command. The name is left justified and padded with blanks on the right. The area contains blanks if RACF checking is not in effect.

Table 131. Command input format

Offset	Offset (decimal)	Name	Length/Alignment	Description
X'0'	0	MSGLL	2	record length
X'2'	2	MSGZZ	2	reserved length
X'4'	4	CRC	1	command recognition character
X'5'	5	CMDDATA	nnn	external subsystem command

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Command exit routine successful. The Command exit routine accepted the command input message.
4	Command exit routine unsuccessful. The Command exit routine rejected the command input message.
20	Should not occur.

Related reference

“Message Service exit routine” on page 374
An external subsystem uses this exit routine when it wants IMS to send a message to an IMS destination (MTO or input terminal).

Commit Continue exit routine

The Commit Continue exit routine provides the second phase of the two-phase commit process.

In other words, the data associated with the current PSB is committed to the database, locks are released, and cleanup is performed. This exit routine is activated after all participating subsystems have voted 'yes' (return code 0 from Commit Prepare exit routine) to the commit prepare request.

Subsections:

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'2'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Commit Continue successful. **Action:** IMS continues normal processing.
4	Commit Continue unsuccessful. ESAP or external subsystem processing of the request failed. **Action:** IMS terminates the application with abend U3046 (the input message is processed; DL/I resources are committed). The control region performs resolve indoubt processing for the recovery token. The dependent region is terminated, which implicitly terminates the dependent region connection to the external subsystem (Signoff and Terminate Identify exit routines are not called). BMP jobs that must be resubmitted resume processing after the commit point.
20	Should not occur.

Related reference

“Commit Prepare exit routine” on page 344
The Commit Prepare exit routine is activated by IMS when an update or non-update transaction reaches a sync point.

“Resolve Indoubt exit routine” on page 358
The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

Commit Prepare exit routine

The Commit Prepare exit routine is activated by IMS when an update or non-update transaction reaches a sync point.

Sync points include:

- Get unique (GU) call to the message queue
- Application-initiated checkpoint
- Application program termination

On return, the exit routine must indicate whether it is prepared to commit all uncommitted changes initiated by the currently scheduled application. The exit routine can indicate whether or not the second phase of the commit process (commit continue) is required. If the transactions associated with the sync point processing are non-update transactions, they do not need to be committed, in which case the exit routine returns with a return code of X'C', requesting that IMS not call the Commit Continue exit routine.

Subsections:

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'2'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
X'00'	Commit Prepare successful. **Action:** IMS continues normal processing.
X'04'	Commit Prepare unsuccessful. The external subsystem is not prepared to perform commit processing at this time. **Action:** • If the application is not terminating, IMS drives the Abort Continue exit routine. An internal ROLB is performed, which returns the input message to the application. • If the sync point was the result of the application terminating, IMS activates the Terminate Thread exit routine with the abort option. The application is terminated with abend U3055, updates are discarded, and the input message is re-enqueued.
X'08'	Commit Prepare unsuccessful. Prepare processing failed in the external subsystem. **Action:** IMS activates the Abort Continue exit routine for all participating subsystems (if the application is not terminating) or the Terminate Thread exit routine with the abort option. The application terminates with abend U3044 and updates are discarded.
X'0C'	Commit Prepare successful for nonupdate transactions. **Action:** IMS continues normal processing but does not call the Commit Continue exit routine. The external subsystem indicated that it is processing nonupdate transactions that do not need to be called for the second phase of commit processing. If the application program is terminating, IMS calls the Terminate Thread exit routine.

Return code	Meaning
X'18'	Commit Prepare unsuccessful. The request was rejected because the recovery token presented by IMS at commit prepare already existed in the external subsystem. One of the following conditions occurred: • Outstanding recovery was not resolved by the Resolve indoubt exit routine, probably due to errors in the external subsystem. • IMS was cold started and the contents of the recovery token occurred once again. **Action:** IMS pseudo abends the application program with abend U3053 and backs out the previous updates. The application is immediately rescheduled. The dependent region connection is reestablished whereupon a new recovery token is presented to the Signon exit routine.
X'20'	Should not occur.

Related reference

“Terminate Thread exit routine” on page 369
The Terminate Thread exit routine disconnects the application from the external subsystem. It is activated when the application program terminates normally.

“Resolve Indoubt exit routine” on page 358
The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

Commit Verify exit routine

IMS calls the Commit Verify exit routine during Get Unique (GU) message processing when the transaction is defined as MODE=MULT.

This kind of transaction allows multiple messages to be processed without an intervening commit action.

IMS calls the exit routine before the next message is dequeued and presented to the application program. The exit routine allows the external subsystem to decide if it can properly process a new message without initiating a commit for the previous message. The external subsystem returns to IMS with a return code that requests that IMS continue with normal MODE=MULT (or CMTMODE(MULT)) processing or initiate a commit action. If a commit action is requested, IMS will initiate the commit action before dequeuing the next message and will terminate the application program with a "QC" status code.

Subsections:

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'3'.
X'4'	4	Address of the EEVT prefix.

Offset (hexadecimal)	Decimal	Content
X'8'	8	Address of the 8-character user ID, left justified and padded with blanks. The user ID is associated with the message that is currently being processed (the next message has not yet been dequeued) and is identical to the one that was presented to the external subsystem at the time IMS last called the Signon exit routine.
X'C'	12	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems. This recovery token is identical to the one that was presented to the external subsystem when IMS last called the Signon exit routine.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Commit Verify processing successful. The external subsystem indicates that it can support MODE=MULT processing without initiating a commit action. **Action:** IMS continues normal MODE=MULT processing. The next message will be dequeued and presented to the application program without initiating a commit action.
4	Commit Verify processing successful. The external subsystem indicates that it cannot support MODE=MULT processing at this time. IMS needs to initiate a commit action. **Action:** IMS terminates the application with a "QC" status and initiates commit processing. Following the commit action, IMS reschedules the application program and the next message is presented for processing.
8	Commit Verify unsuccessful. Commit Verify processing failed in the external subsystem. **Action:** IMS terminates the application program with abend U3044 and discards all updates.
20	Should not occur.

Create Thread exit routine

The Create Thread exit routine is activated by IMS to create a thread to the external subsystem.

Threads can be created only after the TCB that the application runs under has been identified to the external subsystem. A thread is created for each application that makes a request to the external subsystem. The first request by the application program directed at the selected subsystem initiates the activation. Once the thread is created, application requests flow directly to the external subsystem through the Normal Call exit routine.

Subsections:

- "Activating the routine" on page 347
- "Contents of register 15 on return" on page 348

Activating the routine

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'5'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the eight-character application program name, left justified and padded with blanks to the right.
X'C'	12	Address of the eight-character PSB name, left justified and padded on the right with blanks.
X'10'	16	Address of the contents of register 0 in the application save area. When register 0 was saved, it contained the address of the external subsystem-directed parameter list constructed by the language interface.
X'14'	20	Address of a two-character transaction characteristic field. The fields are described from left to right. Byte one contains one of the following: **U** Indicates the transaction was defined by the installation as capable of update. **N** Indicates the transaction was defined by the installation as non-update. Byte two contains one of the following: **S** Indicates the transaction was defined by the installation as mode=single. **M** Indicates the transaction was defined by the installation as mode=multiple.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
X'00'	Create Thread successful. **Action:** IMS continues normal processing.
X'04'	Create Thread unsuccessful. The external subsystem rejected the specified request. **Action:** IMS activates the Subsystem Not Operational exit routine. The return code from the Subsystem Not Operational exit routine determines further processing. Return code 4, coupled with a return code 4 out of the Subsystem Not Operational exit routine, causes an application loop unless the application checks the return code presented by the API.

Return code	Meaning
X'08'	Create Thread temporarily unsuccessful. The external subsystem was unable to complete the request due to the unavailability of a required resource (resource allocation failure). **Action:** IMS terminates the application program with abend U3048.
X'0C'	Create Thread permanently unsuccessful. The request failed in the external subsystem. **Action:** IMS terminates the application program with abend U3045.
X'10'	Create Thread unsuccessful. The request failed in the external subsystem because communications with it are broken. **Action:** IMS terminates the application program with abend U3044.
X'14'	Create Thread unsuccessful. The external subsystem was unable to complete the request due to a definitional conflict. **Action:** IMS terminates the application program with abend U3047.
X'20'	Should not occur.
X'24'	Create Thread unsuccessful. A resource deadlock was detected by the external subsystem. **Action:** IMS terminates the application program with abend U777. All changes are backed out and the application is rescheduled.

Related reference

“Normal Call exit routine” on page 356
The subsystem-supplied Normal Call exit routine is activated by IMS when a subsystem-directed request is made by an application program and a thread to the subsystem is present.

“Subsystem Not Operational exit routine” on page 364
The Subsystem Not Operational exit routine is typically activated when IMS encounters an unusual situation.

Echo exit routine

The Echo exit routine is activated to determine whether IMS can communicate with the external subsystem.

Activation normally occurs after IMS terminates an application program due to an error processing an external subsystem request. The Echo exit routine is expected to send a 'are you there' message or signal to the external subsystem, soliciting a response.

Subsections:

Activating the routine

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Echo successful. The external subsystem responded to the Echo exit routine indicating it is able to continue communication. **Action:** IMS continues normal application scheduling and processing.
4	Echo unsuccessful. The external subsystem has not responded to the Echo exit routine or responded in error. **Action:** IMS PSTOPs the transaction in question.
20	Should not occur.

Related reference

“Resolve Indoubt exit routine” on page 358
The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

Identify exit routine

IMS activates the Identify exit routine to establish a connection from the control region or a dependent region to the external subsystem.

Initial contact from the region to the external subsystem is through this exit routine. (The Identify exit routine is expected to communicate with the external subsystem whereas the Initialization exit routine, if provided, might only perform ESAP initialization and not actually communicate with the external subsystem.) Successful activation of the exit routine (for example, return code 0) is necessary in order for the region to be able to communicate with the external subsystem.

An aspect of the identify concept is the identification of IMS TCBs to the external subsystem. When an IMS TCB terminates abnormally and in some cases when a dependent region terminates normally, IMS does not inform the external subsystem of the termination. The external subsystem should monitor, with z/OS end-of-task (EOT) exit routines, the TCBs for the regions that identify so that it can be notified by z/OS of a termination that was not communicated by IMS.

In the control region and in an MPP- or IFP-dependent region, the Identify exit routine is activated during region initialization processing unless the Initialization exit routine returned with return code 4 (do not identify). The Identify exit routine (control or dependent region) is also activated when the external subsystem activates (through an exit routine) the Subsystem Startup Service supplied by IMS.

IMS passes a notify message on the control region identify request. If the exit routine returns with return code 4 (notify message accepted), IMS waits for the external subsystem to send the notify message before reactivating the exit routine to establish the connection. This return code is intended to be used (optionally) in the case where the external subsystem is not active when IMS attempts to identify.

Related Reading: See *IMS Version 15.2 Communications and Connections* for more information about notify message.

IMS also passes the address of a termination ECB to the control region Identify exit routine. The external subsystem can post this ECB to cause IMS to terminate the connection; for example, when the external subsystem is shutting down.

Related Reading: Refer to "Terminating the external subsystem connection" in *IMS Version 15.2 Communications and Connections* for more information.

Subsections:

Activating the routine from the control region

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates control region environment (control region TCB).

The exit parameter list (EPL) contains:

Offset (hexadecimal)	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'5'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the 4-character external subsystem name.
X'C'	12	Address of the 8-character field containing the IMS system ID (4 characters blank filled to 8 bytes). In an XRF complex, this field contains the RSENAME.
X'10'	16	Address of the notify message area. See *IMS Version 15.2 Communications and Connections*.
X'14'	20	Address of the subsystem termination ECB.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
X'0'	Identify successful. **Action:** IMS performs resolve indoubt processing in the control region.
X'4'	Identify unsuccessful. The external subsystem will send the notify message when it is ready to connect. **Action:** The external subsystem connection is not established. IMS waits for receipt of the notify message before activating the Identify exit routine again. Calling the IMS Subsystem Startup Service after Identify return code 4 does not cause the Identify exit routine to be reactivated.
X'8'	Identify unsuccessful. The notify message was accepted on a previous identify request. **Action:** IMS waits for receipt of the notify message before activating the exit routine again.

Return code	Meaning
X'C'	Identify unsuccessful. The identify process failed, either in the ESAP or the external subsystem. **Action:** If an application is involved, it terminates with abend U3044.
X'20'	Should not occur.

Activating the routine from the dependent region

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'3'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the four-character external subsystem name.
X'C'	12	Address of the IMS system ID.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
X'0'	Identify successful. The dependent region TCB has successfully identified to the external subsystem. **Action:** Signon processing follows.
X'C'	Identify unsuccessful. The identify process failed, either in the ESAP or the external subsystem. **Action:** If an application is involved, it terminates with abend U3044.
X'20'	Should not occur.

Related reference

“Resolve Indoubt exit routine” on page 358
The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

Initialization exit routine

IMS activates the optional Initialization exit routine to allow the ESAP to initialize work areas or control blocks.

IMS activates the optional Initialization exit routine to allow the ESAP to initialize work areas or control blocks in the following instances:

- During the initial stages of establishing a connection from the control or dependent regions. Activation occurs after IMS has constructed its required control blocks as well as the control blocks for the

external subsystem. This action takes place before the control or dependent regions have their respective Identify exit routine activated.

- In a dependent region after an application program abend.

If the Initialization exit routine sets the 'do not identify' return code (return code 4), or if an Initialization exit routine is not supplied, IMS does not automatically perform identify processing for the region. See *IMS Version 15.2 Communications and Connections* for information on how the identify process is eventually performed.

Subsections:

- "Activating the routine from the control region" on page 353
- "Contents of register 15 on return" on page 353
- "Activating the routine from the dependent region" on page 354
- "Contents of register 15 on return" on page 354

Activating the routine from the control region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.
8	8	Address of the 1-byte alphabetic region error option (REO) character defined by the installation. The exit routine should save the error option for future reference when a decision concerning the application is required.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Initialization successful. **Action:** IMS attempts to initiate a connection with the external subsystem by activating the Identify exit routine.
4	Initialization successful. Do not identify to the external subsystem. **Action:** IMS does not perform identify processing during control region initialization. It is now the responsibility of the external subsystem to initiate the connection using the IMS Subsystem Startup Service.
8	Initialization unsuccessful. **Action:** IMS does not initiate a connection to the subsystem. The external subsystem is marked as unstartable. The /START SUBSYS command resets the condition.
20	Should not occur.

Activating the routine from the dependent region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.
8	8	Address of the 1-byte alphabetic region error option character. The region error option is user-defined as part of the SSM.PROCLIB member. The exit routine code should save the error option for future reference when a decision concerning the application is required.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Initialization successful. **Action:** For an MPP or IFP region, IMS initiates a connection to the external subsystem during region initialization. (IMS does not automatically initiate a connection for a BMP region. See the description for return code 4.)
4	Initialization successful. Do not identify to the external subsystem. **Action:** IMS does not automatically initiate a connection for the dependent region. When the region processes the first application call to the external subsystem, the ESAP is expected to activate the IMS Subsystem Startup Service (from the Subsystem Not Operational exit routine. This is always the case for BMP regions (that is, when return code 0 is set). Return code 4 has significance only for MPP and IFP regions.
8	Initialization unsuccessful. **Action:** IMS does not initiate a connection to the subsystem for the life of the execution of this dependent region.
20	Should not occur.

Related reference

“Subsystem Not Operational exit routine” on page 364
The Subsystem Not Operational exit routine is typically activated when IMS encounters an unusual situation.

“Subsystem Startup Service exit routine” on page 376

The IMS Subsystem Startup Service exit routine is activated by an external subsystem exit routine to cause IMS to initiate a connection to the external subsystem.

BPEUXCSV free storage service

The BPEUXCSV free storage service is used to release storage that was previously obtained with the BPEUXCSV get storage service.

The free storage service is similar to the z/OS FREEMAIN service. It must be used only to release storage obtained with the get storage service. It should not be used to release storage that was obtained using any other method (such as GETMAIN).

Service Code: BPEUXCSV_FREESTG

PARMS format:

PARMS=(*address,length,sp*) or PARMS=(*address,length,sp,key*)

address
The address of the first byte of storage being released.

length
The number of bytes of the storage being released.

sp
The subpool of the storage being released. This subpool must be the same as the subpool that was specified when the storage was obtained.

key
The storage key of the storage being released. *key* is the optional parameter. If coded, it indicates the storage key of the storage being freed. If *key* is omitted, the storage must be key 7 storage.

The value passed for the *key* parameter must be sixteen times the actual key value. For example, if you were freeing key 2 storage, you would specify a value of X'20' for the *key* parameter.

Note: The *key* parameter *only* applies to subpools where KEY= applies on the z/OS FREEMAIN macro (for example, subpool 229). It is ignored for all other subpools.

Output: Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, R0 contains the reason code; the following table lists the reason codes, including the symbol, its value, and a description.

Table 132. Free storage service return codes

Symbol	Value	Description
BPEUXCSV_FREESTG_RCSP	X'04'	An invalid or unsupported subpool was specified. Either the subpool is not supported by z/OS, or it is a common subpool.
BPEUXCSV_FREESTG_RCLV	X'08'	A zero or negative storage length was specified.
BPEUXCSV_FREESTG_RCADDR	X'0C'	A zero storage address was specified.
BPEUXCSV_FREESTG_RCSTG	X'10'	The service was unable to free the requested storage.
BPEUXCSV_FREESTG_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.

Example:

This example shows how to free STGLEN bytes starting at the byte at label MYSTG in subpool 129. STGLEN is an EQU for the number of bytes to free, and MYSTG is the label on the first byte of the area to free (*not* the label on a word pointing to the area).

```
          BPEUXCSV SERVICECODE=BPEUXCSV_FREESTG,                        X
                PARMS=(MYSTG,STGLEN,129),                               X
                TOKEN=UXPL_CSTOKENP,                                    X
                SL=(1)
```

Normal Call exit routine

The subsystem-supplied Normal Call exit routine is activated by IMS when a subsystem-directed request is made by an application program and a thread to the subsystem is present.

The subsystem-supplied Normal Call exit routine is activated by IMS when a subsystem-directed request is made by an application program and a thread to the subsystem is present. It is the responsibility of the Normal Call exit routine to:

- Communicate normal call and data to the external subsystem.
- Handle responses.
- Supply status codes to the application program.

Subsections:

- "Activating the routine" on page 356
- "Contents of register 15 on return" on page 357

Activating the routine

The exit routine is activated in the caller's key. The caller is IMS, an application program, or an external subsystem-supplied exit routine, and is either authorized or unauthorized. If the caller is authorized, the exit routine is activated in key 7, supervisor state. If the caller is unauthorized, the exit routine is activated in key 8, problem program state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset	Offset (decimal)	Content
X'0'	0	Address of the parameter count field. The count field contains the value F'6'.
X'4'	4	Address of the EEVT prefix.
X'8'	8	Address of the contents of register 0 in the application save area. At this time register 0 contains the address of the external subsystem-directed parameter list as constructed by the language interface.
X'C'	12	Address of the contents of register 1 in the application save area. Register 1 contains the address of the application parameter list.
X'10'	16	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Offset	Offset (decimal)	Content
X'14'	20	Address of a one-character field which identifies the authorization state: **A** The caller is authorized and the exit routine is activated in key seven, supervisor state. **U** The caller is unauthorized and the exit routine is activated in key eight, problem program state.
X'18'	24	Address of a 12-word buffer area provided for specific language function calls. External subsystems that require IMS to call internal exit routines for post-normal call processing can use this buffer to pass data to the internal exit routine. If post-normal call processing is required, IMS passes the address of the buffer to the associated internal exit routine. If post-normal call processing is **not** required, the external subsystem should not use this parameter. For more information, see return code 12. **Restriction:** The use of this address field is restricted to those external subsystems that have negotiated the definition and use of an internal exit routine for post-normal call processing. Requests for this support should be made through the IBM User Requirements Process, which includes GUIDE, SHARE, and vendor requirements processing.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
X'0'	Normal Call successful. **Action:** IMS continues normal processing.
X'4'	Normal Call unsuccessful. A resource deadlock is detected by the external subsystem. **Action:** IMS terminates the application program with abend U777. All changes are backed out and the application is rescheduled.
X'8'	Normal Call unsuccessful. A failure in the external subsystem occurred while processing the request. **Action:** IMS terminates the application with abend U3044.
X'C'	Normal Call successful. The external subsystem requested the scheduling of an associated internal exit routine for post-normal call processing. IMS calls the internal exit routine before returning control to the application program. **Action:** If there is an internal exit routine associated with the language function call, IMS calls the associated internal exit routine and passes the address of the buffer. If there are no internal exit routines associated with it, IMS ignores the request and performs the processing associated with return code 0. The actual interface to an internal exit routine is unique to that routine and depends on the type of external subsystem. The external subsystem-specific interfaces are not documented.
X'20'	Should not occur.

Resolve Indoubt exit routine

The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems. IMS, as the recovery coordinator, always calls this exit routine after successful completion of the identify process. IMS indicates in the EPL whether or not recovery must take place for the units of work in question.

The Resolve Indoubt exit routine is activated once for each outstanding IMS recovery token. It is called after the Echo exit routine. The external subsystem directs IMS to save or destroy the recovery token. More information on exit routine responses is in "Contents of register 15 on return" on page 359.

The Resolve Indoubt exit routine has the option of allowing the two subsystems to continue communication with or without all recovery tokens resolved. If communication continues and outstanding recovery tokens exist, an authorized operator can direct IMS to delete its recovery tokens using the /CHANGE command.

Related Reading: Refer to *IMS Version 15.2 Commands, Volume 1: IMS Commands A-M* for information on the /CHANGE command.

If the Resolve Indoubt exit routine address is not present in the EEVT and outstanding recovery tokens do not exist, IMS allows the connection process to continue. However, if a recovery token does exist, IMS terminates the connection and informs the MTO with message DFS3602I.

The Resolve Indoubt exit routine is also activated after the abend of an application program that had a connection (thread) to the external subsystem.

Subsections:

Activating the routine

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'4'.
4	4	Address of the EEVT prefix.

Offset (hexadecimal)	Decimal	Content
8	8	Address of a two-character field: • Byte 1 contains an indicator, either 'C' or 'W', on the first activation of the Resolve Indoubt exit routine during the current IMS execution. On subsequent activations, the byte contains binary zeroes. (For example, if the external subsystem terminates and restarts while IMS remains active, when the connection is reestablished, the byte will contain binary zeroes.) **C** Indicates IMS was cold started. All subsequent fields in the parameter list contain binary zeroes. **W** Indicates IMS was warm started. • Byte 2 is set to 'L' after the last recovery token of the current sequence is processed. For all other activations, the byte is set to binary zeroes. **L** Indicates either that there are no recovery tokens to be resolved, or that all recovery tokens that were to be resolved at this time have been processed. If 'L' is not set on an activation of the exit routine, the exit routine should expect to be activated one or more times (once per recovery token) until 'L' is set. A recovery token is not passed on the last ('L') activation. When 'C' is set in byte one, 'L' is always set because IMS does not save recovery information across a cold start.
C	12	Address of a two-character field indicating whether the recovery unit should be aborted or committed. **CO** Commit **AB** Abort
10	16	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Resolve Indoubt successful. **Action:** IMS continues normal processing. The recovery token is destroyed.

Return code	Meaning
4	Resolve Indoubt unsuccessful. Return code 4 should be set when the external subsystem chooses not to resolve (commit or abort as directed) the unit of work at this time and expects IMS to pass the recovery token for resolve indoubt processing at a later time. This return code is not intended for the case where resources have become inconsistent or the possibility exists (see return code C). **Action:** IMS saves the recovery token. The connection status remains unchanged. IMS assumes that the indicated unit of work is indoubt status in the external subsystem (for example, resources have not become inconsistent). The saved recovery token will be included in the resolve indoubt processing for the next establishment of the connection. IMS does not inform the installation that the unit of work was not resolved.
8	Resolve Indoubt unsuccessful. Return code 8 can be used when the external subsystem chooses not to resolve the unit of work during exit processing but saves the commit direction so that IMS does not need to save the recovery token. This return code is not intended for the case where resources have become inconsistent (see return code C). Return code 8 might be used when the indicated unit of work is not in indoubt status in the external subsystem but resource consistency is not jeopardized, however, **caution is advised**. External subsystem-specific processing that is not coordinated with IMS can result in IMS holding a recovery token in indoubt status when the unit of work is not indoubt in the external subsystem (for example, external subsystem "cold start", or manual recovery of a unit of work by the installation if allowed by the external subsystem). If the external subsystem can determine that its prior resolution of a unit of work (explicit or implicit) agrees with the commit direction passed to the exit routine, return code 8 can be set; otherwise, return code C should be set. **Action:** IMS destroys the recovery token. The connection status remains unchanged (IMS assumes that resource consistency is maintained). The IMS action is the same as for return code 0. Setting return code 8 allows for an audit trail of the "not-quite-normal" cases.
C	Resolve Indoubt unsuccessful. Return code C should be used when resources in IMS and the external subsystem have become inconsistent or when the possibility exists. It is intended to be used, for example, when the recovery token passed to the exit does not exist (is unknown) in the external subsystem, or when the external subsystem has indoubt recovery tokens remaining when IMS has finished processing its indoubt list. The external subsystem should take appropriate additional action to maintain integrity and assist the installation in resolving resource inconsistencies. **Action:** IMS terminates the connection and saves all remaining recovery tokens. IMS also issues message DFS3602I to notify the installation of a resource problem. • If a recovery token was passed on the exit routine activation (for example, **L** was not set), IMS terminates the connection to the external subsystem. The recovery token passed and all remaining recovery tokens are saved. • If this is the last activation (**L** was set), the connection status is unchanged. Dependent regions are allowed to connect to the external subsystem.
20	Should not occur.

Signoff exit routine

The Signoff exit routine is activated by IMS during shutdown or termination of IMS subsystems.

The Signoff exit routine is activated by IMS when:

- IMS is shutting down.

- The external subsystem activates the IMS Subsystem Termination Service.
- The subsystem termination notification ECB is posted.
- The /STOP SUBSYS command is entered and IMS is terminating all the subsystem connections.

IMS attempts to activate the exit routine as part of an orderly/catastrophic shutdown process of the two subsystems.

- "Activating the routine" on page 361
- "Contents of register 15 on return" on page 361

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Signoff successful.
8	Signoff unsuccessful. The ESAP or external subsystem processing of the request failed. **Action:** IMS terminates the dependent region connection with the external subsystem, allowing other dependents to continue processing.
20	Should not occur.

Signon exit routine

The Signon exit routine informs the external subsystem of the user ID associated with the transaction input message.

The user ID can be:

- The inputting LTERM name (if the terminal user is not signed on)
- The RACF/user-authorized user ID associated with a non-message driven BMP or CPI-C application
- The PSB name specified on the job card
- The ID of the terminal user

The following table lists, in search order, the fields that IMS will check when it searches for a user ID. For each field, it lists the criteria that IMS uses to validate the user ID. When IMS finds a valid user ID, IMS extracts the ID and passes it to the Signon exit routine.

Table 133. Determining the signon user ID

Type of application	Field	Criteria for authorized user ID
CPI-C	1. RACF user ID if the accessor environment element (ACEE) is cloned in the dependent region	Value passed without validation
	2. PSTBUSER	The value is not binary zeroes or blanks
	3. PSTUSID	The value is not blanks
	4. PSTSYMBO	The value is not blanks
	5. PDIRSYM	Value passed without validation
• Message-driven BMP that has done a Get Unique call • IFP that has done a Get Unique call • MPP	1. PSTUSID	The value is not blanks
	2. PSTSYMBO	The value is not blanks
	3. PSTBUSER	The value is not binary zeroes or blanks
	4. PDIRSYM	Value passed without validation
• Non-message-driven BMP • Message-driven BMP that has not done a Get Unique call • IFP that has not done a Get Unique call	1. PSTBUSER	The value is not binary zeroes or blanks
	2. PDIRSYM	Value passed without validation

When a dependent region connection is initially established, the Signon exit routine is activated before a thread is created by the Create Thread exit routine. All subsequent requests result in the exit routine being activated after a thread is created. For example, Signon is activated for each message processed during a single scheduling, whether or not the messages are separated by commit processing.

- "Activating the routine" on page 362
- "Contents of register 15 on return" on page 363

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'7'.
4	4	Address of the EEVT prefix.
8	8	Address of the eight-character user ID, left justified and padded on the right with blanks.
C	12	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Offset (hexadecimal)	Decimal	Content
10	16	Address of the 8-byte RACF group name for the user ID that entered the transaction. The name is left justified and padded with blanks on the right. The area contains blanks if RACF checking is not in effect.
14	20	Address of the field containing the performance block token for z/OS workload management support.
18	24	Address of the XID token associated with this transaction. The XID token identifies participants in a Distributed Syncpoint environment.
1C	28	Address of the ACEE for this transaction instance or 0.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Signon successful. **Action:** IMS continues normal processing.
4	Signon unsuccessful. The external subsystem rejected the specified request. **Action:** IMS activates the Subsystem Not Operational exit routine. The return code from the Subsystem Not Operational exit routine determines further processing.
8	Signon temporarily unsuccessful. The external subsystem was unable to complete the request due to the unavailability of a required resource (resource allocation failure). **Action:** IMS terminates the application program with abend U3048.
C	Signon permanently unsuccessful. The request failed in the external subsystem. **Action:** IMS terminates the application program with abend U3045.
10	Signon unsuccessful. The request failed in the external subsystem because communications with it are broken. **Action:** IMS terminates the application program with abend U3044.
14	Signon unsuccessful. The external subsystem was unable to complete the request due to a resource definitional conflict between the two subsystems. **Action:** IMS terminates the application program with abend U3047.
18,1C	Signon unsuccessful. The request was rejected because the recovery token presented by IMS at signon already exists in the external subsystem. One of save following conditions occurred: • Outstanding recovery was not resolved by the Resolve Indoubt exit routine, probably due to errors in the external subsystem. • IMS was cold started and the contents of the recovery token occurred once again. **Action:** IMS pseudo abends the application program with abend U3053 and backs out the previous updates. The application is immediately rescheduled. The dependent region connection is re-established whereupon a new recovery token is presented to the Signon exit routine.

Return code	Meaning
20	Should not occur.

Related reference

"Subsystem Not Operational exit routine" on page 364
The Subsystem Not Operational exit routine is typically activated when IMS encounters an unusual situation.

"Resolve Indoubt exit routine" on page 358
The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

Subsystem Not Operational exit routine

The Subsystem Not Operational exit routine is typically activated when IMS encounters an unusual situation.

The Subsystem Not Operational exit routine is viewed as a utility type of exit routine. IMS activates this exit routine when:

- An application program directs a request to the external subsystem and a connection does not exist. The Subsystem Not Operational exit routine can activate the IMS Subsystem Startup Service exit routine to initiate a connection.
- Return code 4 is returned from the Signon or Create Thread exit routines.
- The external subsystem tells IMS it is quiescing before creation of a thread.

Subsections:

Activating the routine

The exit routine is activated in the caller's key. The caller is IMS, an application program, or an external subsystem-supplied exit routine, and is either authorized or unauthorized. If the caller is authorized, the exit routine is activated in key seven, supervisor state. If the caller is unauthorized, the exit routine is activated in key eight, problem program state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'10'.
4	4	Address of the EEVT prefix.
8	8	Address of the contents of register 0 in the application program save area. At this time register 0 contains the address of the external subsystem-directed parameter list as constructed by the language interface.
C	12	Address of the contents of register 1 in the application program save area. Register 1 contains the address of the application parameter list.

Offset (hexadecimal)	Decimal	Content
10	16	Address of a one-character information field. The contents indicate why the Subsystem Not Operational exit routine is being activated. The field contains: **A** Return code 4 was returned by the Signon or Create Thread exit routines. **C** The IMS control region has not identified to the external subsystem. This condition was discovered when an application directed a request to the subsystem. The Subsystem Not Available exit routine can activate the IMS Subsystem Startup Service to initiate a connection. **D** An application issued a call to the external subsystem but the dependent region has not identified. The Subsystem Not Operational exit routine can activate the IMS Subsystem Startup Service to initiate a connection. **Q** The external subsystem notified IMS that it is terminating in a quiesce fashion. Prior to the creation of a thread is the only interval where 'Q' is passed to the external subsystem (applicable mainly when the region error option (REO) is an 'R'). **T** The external subsystem notified IMS that it is either abnormally terminating or terminating in a quick fashion. It is highly likely that a subsystem-directed request will fail. IMS notifies the external subsystem when servicing a subsystem request (such as Create Thread).
14	20	Address of a one-character default application option field. This field contains the region error option (REO) defined by the installation in the external subsystem PROCLIB member to take effect in the event an application issues a subsystem-directed request when a complete authorized connection does not exist. This field is always valid. If an option is not specified, IMS uses its elected default ('R').
18	24	Address of a four-byte character format field containing the name of the subsystem associated with the request. The name is left justified and padded with blanks.
1C	28	Address of a fullword where the Subsystem Not Operational exit routine optionally returns a z/OS format abend code.
20	32	Address of the application program name scheduled at this time. The name is assumed to be left justified and padded with blanks to the left. The maximum length is eight bytes.
24	36	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Offset (hexadecimal)	Decimal	Content
28	40	Address of a one-character field which identifies the authorization state: **A** The caller is authorized and the exit routine is activated in key seven, supervisor state. **U** The caller is unauthorized and the exit routine is activated in key eight, problem program state.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Subsystem Not Operational call successful. **Action:** IMS continues normal processing. Normal processing in this case means activating the Signon and Create Thread exit routines to complete the external subsystem connection.
4	Subsystem Not Operational call successful/unsuccessful. This exit routine elected to pass status feedback to the application. **Action:** IMS returns control to the application program. The application is permitted to continue running. A loop between the Create Thread exit routine and the Subsystem Not Operational exit routine might result if the application program does not check for nonzero return codes from the API.
8	Subsystem Not Operational call unsuccessful. **Action:** IMS terminates the application program with abend U3044. The transaction input is saved and all uncommitted changes are backed out. The dependent region remains available for application processing.
C	Subsystem Not Operational call unsuccessful. **Action:** IMS terminates the application program with abend U3047 and discards the input. The dependent region remains available for application processing.
10	Subsystem Not Operational call unsuccessful. **Action:** IMS uses the z/OS format abend code returned by this exit routine to abend the application.
20	Should not occur.

Related reference

“Signoff exit routine” on page 360
The Signoff exit routine is activated by IMS during shutdown or termination of IMS subsystems.

“Create Thread exit routine” on page 347

The Create Thread exit routine is activated by IMS to create a thread to the external subsystem.

Subsystem Termination exit routine

The Subsystem Termination exit routine is activated in the control region when IMS or the external subsystem terminate; activation follows the Terminate Identify exit routine. The Subsystem Termination exit routine is used for cleaning up work areas or freeing memory.

The Subsystem Termination exit routine should execute in parallel with normal and abnormal IMS or external subsystem termination processing. External subsystem termination is recognized when the subsystem posts the termination ECB.

- "Activating the routine" on page 367
- "Contents of register 15 on return" on page 368

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.
8	8	Address of a 1-byte character format field indicating the reason for subsystem termination. The field contains one of the following: **A** IMS is shutting down in a normal fashion (/CHECKPOINT FREEZE). IMS makes sure new connections are not established and permits existing ones to terminate normally. **B** IMS is shutting down abnormally (abend). Some abend conditions might prohibit the activation of this exit routine. **C** The external subsystem notified IMS that it is terminating in a quiesce fashion. IMS makes sure new connections are not established and permits existing ones to terminate normally. **D** The external subsystem notified IMS that it is terminating abnormally (catastrophic). IMS makes sure new connections are not attempted and terminates existing ones. **E** The connection between the subsystems is being quiesced by IMS. IMS is not shutting down but remains available. The termination of the connection is the result of the /STOP command, a bad return code from an exit routine, or a required exit routine missing. **F** The connection between the subsystems is being terminated because the IMS Subsystem Termination Service exit routine was activated by an external subsystem exit routine.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Subsystem Termination successful. **Action:** IMS continues subsystem termination.
8	Subsystem Termination unsuccessful. The ESAP or the external subsystem encountered a failure in processing the termination notification. **Action:** IMS continues termination processing. Future connection requests are honored.
20	Should not occur.

Terminate Identify exit routine

The Terminate Identify exit routine is activated by IMS to terminate the hierarchical structure established between control regions and their dependent regions.

Each IMS region that has an external subsystem connection must first identify to the subsystem. Identify must first be complete for the control region before any dependent regions identify. This hierarchical structure allows the control region to act as recovery coordinator for the dependent regions. If a dependent region were to fail, the control region intervenes and instructs the external subsystem to commit or abort the dependent region units of work.

The Terminate Identify exit routine is activated by IMS to terminate this hierarchical structure when:

- IMS is shutting down.
- The subsystem activates the IMS Subsystem Termination Service.
- The subsystem termination notification ECB is posted.
- The /STOP SUBSYS command is entered.

Activation is part of an orderly/catastrophic shutdown or disconnecting process of the two subsystems.

Subsections:

- "Activating the routine from the control region" on page 368
- "Activating the routine from the dependent region" on page 369
- "Contents of register 15 on return" on page 369

Activating the routine from the control region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Activating the routine from the dependent region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Terminate Identify successful. **Action:** IMS continues termination processing.
8	Terminate Identify unsuccessful. The ESAP or external subsystem processing of the request failed. **Action:** IMS continues connection termination processing without affecting other dependent region connections.
20	Should not occur.

Terminate Thread exit routine

The Terminate Thread exit routine disconnects the application from the external subsystem. It is activated when the application program terminates normally.

The second phase of the commit processing at application termination is done through the Terminate Thread exit routine. Therefore, a commit option is specified on the activation.

- "Activating the routine" on page 369
- "Contents of register 15 on return" on page 370

Activating the routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EVVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Hexadecimal	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'3'.
4	4	Address of the EEVT prefix.

Hexadecimal	Decimal	Content
8	8	Address of the four-byte commit option character string: **COMM** Commit and terminate the thread. (Normal application termination is an example of when this option is set). **ABRT** Abort and terminate the thread. **DEAL** Deallocate resources with terminate thread (nothing to commit).
C	12	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of register 15 on return

Processing of return codes received from the exit routine:

Return code	Meaning
0	Terminate Thread successful. **Action:** IMS continues normal processing.
4	Terminate Thread unsuccessful. The ESAP or external subsystem processing of the request failed. **Action:** IMS terminates the application with an abend. The dependent region connection to the external subsystem is also terminated. Resolve indoubt processing for the recovery token is performed in the control region. • For the COMM (commit) option: The application is terminated with abend U3046 (the input message is processed; DL/I resources are committed). BMP jobs must be resubmitted; they resume processing after the commit point. • For the ABRT (abort) option: The application is terminated with abend U3045 (the input message is deleted; DL/I resources are backed out). BMP jobs must be resubmitted; they resume processing at the prior commit point.
8	Terminate Thread unsuccessful. The Terminate Thread exit routine has either detected an error with the request information or considers the request invalid at this time. This return code should only be used when the commit option character string is 'DE'. **Action:** IMS continues normal processing.
20	Should not occur.

Related reference

“Resolve Indoubt exit routine” on page 358
The Resolve Indoubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems.

ESAF synchronous exit routines

The External Subsystem Attach Facility (ESAF) can activate IMS-supplied synchronous exit routines to provide prescribed IMS services.

External subsystem exit routines can request IMS to:

- Write a log record on the IMS log (Log Service exit routine).
- Send a message to an IMS destination (Message Service exit routine).
- Initiate a connection (Subsystem Startup Service exit routine).
- Terminate a connection (Subsystem Termination Service exit routine).

External subsystem exit routines are organized alphabetically.

General system service exit routine interface

To activate an IMS system service exit routine, the ESAP loads the address of an IMS service router module from EEVPESGL in the EEVTP control block and branches to it. The ESAP passes required parameters using an exit routine parameter list (EPL) in the same general format as the EPL passed to external subsystem exit routines. Each exit routine has a unique function code. The address of the function code defined for the exit routine being activated must be supplied in the EPL. If a function code address is not passed, the invalid parameter list return code, X'20', is returned to the caller.

On entry, the system service exit routine saves all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of exit parameter list (EPL)
13	Address of save area
14	Return address
15	Address obtained from EEVPSEGL in the EEVTP

Before returning to the ESAP, the system service exit routine restores all registers except register 15, which contains a return code. The parameters and return codes for each system service exit routine are described in the following topics.

The storage key of the save area should be that of the caller, for example, key 8 for the Normal Call exit routine; key 7 for the Subsystem Termination exit routine.

The key of the storage passed to the system service exit routines (Log exit routine, Message exit routine) must be the same as the caller's. For example, if the Identify exit routine wants to place data on the IMS log, that data must exist in storage obtained while the Identify exit routine was running in key 7 before calling the Log exit routine.

IMS system service exit routines should not be activated from TCBs attached by the ESAP. The system service modules expect to be activated under an IMS internal structure that is only available under IMS TCBs.

PDSE resource restrictions

The following restrictions, which are associated with extended partitioned data sets (PDSE), apply to resources (tables and exit routines) that are associated with an external subsystem:

- All executable code, such as exit routines, must have link attributes that are reentrant. These subsystem exit routines must take appropriate actions to prevent access errors.
- Non-executable tables are loaded in the TCB key, or key 0.

Before IMS loads a subsystem resource, IMS locates the resource and determines the type of data set that holds the resource. If the data set type is a PDS, IMS manages the key and subpool of the resource. If the data set is a PDSE, the linkage attributes of the resource determine the key and subpool of that resource.

Non-reentrant code that resides on a PDSE is loaded into TCBKEY(8-15), fetch-protected storage. When IMS gives control to these routines, it does so in IMSKEY(7), which causes fetch-protection errors.

Reentrant code that resides on a PDSE is loaded into KEY0, non-fetch-protected storage. This does not cause fetch-protection errors.

Non-executable tables that reside on a PDSE and are linked as non-reentrant must be referenced in TCBKEY. Otherwise, fetch-protection errors also occur. By linking these tables as reentrant, these errors are prevented.

Log Service exit routine

An external subsystem uses this exit routine when it wants IMS to write a log record.

IMS reserves log record type X'55' for external subsystem usage. The exit routine does not accept any other log record types.

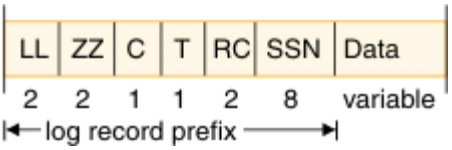
Subsections:

- "Activating the routine" on page 372
- "Contents of register 15 on return" on page 373

Activating the routine

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field must contain the value F'3'.
4	4	Address of the 1-character logging service function code, X'16'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.

Offset (hexadecimal)	Decimal	Content
C	12	Address of the log record area. The exact contents are written to the log. IMS does not alter this area. The log record must be in the following format:  The log record prefix must contain the LL, ZZ, and C fields as follows: **LL** A 2-byte field that must contain the total length of the record. The total length (prefix + data) cannot exceed the logical record length (LRECL) for the system log data set minus 4 bytes. **ZZ** A 2-byte field that must contain binary zeroes. **C** A 1-byte field that must contain the log record type, X'55'. **Recommendation:** The remaining prefix fields are suggested to allow log records for a particular external subsystem to be identified. IMS does not check the contents of the fields. **T** A 1-byte field that contains a system or function type. **RC** A 2-byte field that contains a reason code. **SSN** An 8-byte field that contains the external subsystem name.

Contents of register 15 on return

Register 15 contains one of the following return codes on return:

Return code	Meaning
0	Log request successful. The log record passed by the caller was written to the IMS log.
4	Log request unsuccessful. A failure (or error) was encountered in attempting to process the log request. The record was not logged.
8	Log request unsuccessful. The caller attempted to log an invalid log record type. Only type X'55' log records are accepted by the Log exit routine. The record was not logged.
20	Log request unsuccessful. Invalid data was detected in the exit parameter list. The record was not logged. Possible errors: • Invalid log record address—the EPL contains zeroes or a negative value. • Invalid log record length—the length field contains zeroes or a negative value.

Message Service exit routine

An external subsystem uses this exit routine when it wants IMS to send a message to an IMS destination (MTO or input terminal).

Two types of messages are accepted by the exit routine: preedit (prebuilt) and key call (message number).

Preedit messages are assumed to be formatted by the caller. Single or multiple segment preedit messages are accepted. The address of the message is passed to the exit routine.

On a key call, the address of the message number is passed. The message itself is supplied in the user message table.

Subsections:

- "Activating the routine" on page 374
- "Contents of register 15 on return" on page 376

Activating the routine

The EPL contains:

Offset	Decimal	Content
X'0'	0	Address of the parameter count field. The count field must contain the value F'5'.
X'4'	4	Address of the 1-character message service function code, X'15'. If an address is not present, return code X'20' is returned to the caller.
X'8'	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.
X'C'	12	Address of the message area for preedit messages; address of the message number for key call (user) messages. IMS assumes the message number to be a two-byte binary value. IMS does not alter the message area.
X'10'	16	Address of the 8-byte alphanumeric destination name (that is, the LTERM name). If this field contains binary zeroes, the default message destination is the IMS master terminal. The name is left justified and padded with blanks on the right.

Offset	Decimal	Content
X'14'	16	Address of the 2-character message type indicators. The first character indicates the message type. The second character indicates system versus pageable message. The supported message types in character format are: **S** Message is a single segment and can be pre-edited. **M** Message might consist of a variable number of segments, each segment is prefixed by its own length. The entire message is prefixed by the entire message length. **U** Indicates a user message is sent. When specified, IMS expects the address of a message number passed in the input parameter list. The message is assumed to be present in the IMS user message table. The second character indicates to IMS where the message is to be placed on the master terminal in the event the z/OS split screen master terminal format is used. Traditionally, the top part of the screen is reserved for unsolicited output such as I/O error messages and immediate command response message (that is, command complete, command in progress, invalid keyword or parameter). The lower portion of the screen is where display output (/DISPLAY LTERM ALL) is directed. This allows the MTO to page through the output message using the PA keys. If the split screen master terminal format is not being used, output is displayed, taking advantage of the entire screen. The supported screen format indicators in character format are: **S** Indicates the message being sent is to be placed on the top portion of the screen (system area) **P** Indicates the message being sent is to be placed on the lower portion of the screen (pageable area, typically used for /DISPLAY command output) The format for single segment messages (message type **S**) is: OFFSET LENGTH/ HEX DEC NAME ALIGNMENT DESCRIPTION 0 0 MSGLL 2 record length 2 2 MSGZZ 2 reserved bytes 4 4 MSGDATA nnn substance of message The format for multiple segment messages (message type **M**) is: OFFSET LENGTH/ HEX DEC NAME ALIGNMENT DESCRIPTION 0 0 MSGLL 2 total record length 2 2 MSGZZ 2 reserved bytes 4 4 MSGLL 2 message segment length 6 6 MSGZZ 2 reserved bytes 8 8 MSGDATA 10 first segment of message 12 18 MSGLL 2 message segment length 14 20 MSGZZ 2 reserved bytes 16 22 MSGDATA 20 second segment of message

Contents of register 15 on return

Return code	Meaning
0	Message request successful. The message was enqueued to the master terminal or the specified destination.
4	Message request unsuccessful. The message did not pass validity checking. The message was not sent.
8	Message request unsuccessful. A failure occurred in processing the message request. The message was not sent.
20	Message request unsuccessful. Invalid data was detected in the exit parameter list (that is, invalid destination name specified). The message was not sent. Possible errors: • Invalid EEVTP address—the EPL contains zeroes. • Invalid destination name—the destination name area contains blanks. • Invalid message type.

Related reference

"User Message table (DFSCMTU0)" on page 475
You can create your own messages and list them in the User Message table (DFSCMTU0).

Subsystem Startup Service exit routine

The IMS Subsystem Startup Service exit routine is activated by an external subsystem exit routine to cause IMS to initiate a connection to the external subsystem.

If the control region identify or an MPP or IFP dependent region identify was deferred (for example, not done automatically during region initialization processing), this service is to be used to establish the necessary connection. It can also be used to establish the connection for a BMP dependent region. The external subsystem activates the service from the Subsystem Not Operational exit routine in the dependent region (key 8) when the first application call to the external subsystem is processed in the dependent region.

Related Reading: See *IMS Version 15.2 Messages and Codes, Volume 2: Non-DFS Messages* for more information about the external subsystem connection.

If the control region connection has not been established, the Startup Service exit routine activates the control region identify process before activating the dependent region identify process. The startup exit routine activation fails if the notify message passed to the control region Identify exit routine (on a previous activation) was accepted by the exit routine but the external subsystem has not sent the message to IMS to indicate that it is ready to establish a connection.

Subsections:

Activating the routine

The EPL contains:

Offset (hexadecimal)	Decimal	Content
0	0	Address of the parameter count field. The count field must contain the value F'2'.

Offset (hexadecimal)	Decimal	Content
4	4	Address of the one-character startup service function code, X'17'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.

Contents of register 15 on return

Register 15 contains one of the following return codes on return:

Return code	Meaning
0	Subsystem connection successful. The identify has been performed. IMS continues normal processing.
4	Subsystem connection unsuccessful. The identify process initiated by IMS resulted in the notify message being queued to the external subsystem. However, the subsystem has not yet been started. IMS waits for the receipt of the notify message before continuing the connection process.
8	Subsystem connection unsuccessful. The control region identify was previously attempted, at which time the Identify exit routine accepted the notify message. IMS continues to wait for the external subsystem to send the notify message before reactivating the exit routine to establish the control region connection.
C	Subsystem connection unsuccessful. The connection attempt failed either in the ESAP or in the external subsystem.
10	Subsystem identify unsuccessful. IMS is shutting down.
14	Subsystem connection unsuccessful. IMS is notified that the external subsystem is terminating either in a quiesce or catastrophic fashion.
18	Subsystem connection unsuccessful. During the connection initialization process, the external subsystem Initialization exit routine was activated. The exit routine responded with the `never connect` return code (return code 8).
1C	Subsystem connection unsuccessful. Resources required to process the connection request were unavailable. When this condition exists, IMS sends message DFS3620I to the MTO indicating the resource type. When a required exit routine is missing, message DFS3608 is also sent to the MTO indicating the exit routine name. Refer to *IMS Version 15.2 Messages and Codes, Volume 2: Non-DFS Messages* for more information on these messages.
20	Subsystem connection unsuccessful. Invalid data was detected in the exit parameter list. IMS did not initiate the startup (identify) process.
24	Subsystem connection unsuccessful. The startup request is rejected because an external subsystem exit activated the IMS Terminate Service exit routine. IMS is still in the process of terminating the connection between the two subsystems. Additional dependent region connections are prohibited while in this state.
28	Subsystem connection unsuccessful. The startup request is rejected because IMS is terminating the connection due to a /STOP SUBSYS command entered. Additional dependent region connections are prohibited while in this state.

Subsystem Termination Service exit routine

This IMS-provided exit routine prohibits new connections to the external subsystem but allows the existing connections to terminate normally (quiesce). When all dependent region connections are terminated, the control region connection is terminated.

The Subsystem Termination exit routine is activated by the ESAP, possibly when a subsystem termination command is intercepted by the ESAP exit routine. The intercepting routine is available whether or not the subsystem is attached and running in user key.

To perform the necessary processing, the Subsystem Termination exit routine switches to key 7 supervisor state.

Subsections:

- "Activating the routine" on page 378
- "Contents of register 15 on return" on page 378

Activating the routine

The EPL contains:

Hexadecimal	Decimal	Content
0	0	Address of the parameter count field. The count field must contain the value F'2'.
4	4	Address of the one-character termination service function code, X'18'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.

Contents of register 15 on return

Register 15 contains one of the following return codes on return:

Return code	Meaning
0	Subsystem termination successful. IMS terminates subsystem connections in all dependent regions. When the dependent region connections are all terminated, the control region connection is terminated.
4	Subsystem termination unsuccessful. A failure (or error) was encountered in attempting to process the termination request. IMS terminates the connection to the subsystem as stated under return code 0.
20	Subsystem termination unsuccessful. Invalid data was detected in the exit parameter list. IMS did not initiate termination processing.

IMS Command Language Modification facility (DFSCKWD0)

Use the IMS Command Language Modification facility (DFSCKWD0) to modify the command keyword table.

- "About this facility" on page 378
- "Changing the table" on page 379
- "Error messages" on page 380

About this facility

Several reasons exist for altering the keyword table. For example, you might want to tailor the keywords and synonyms to satisfy unique requirements. A new keyword or keyword synonym in a new IMS release

can conflict with a name already assigned by your installation to a resource such as an LTERM or a transaction. If a new keyword "ABC" is introduced and you already have an LTERM with the name "ABC", you can change the keyword name to "ABCDEFG" and remove the source of the conflict. If the source of the conflict is the new keyword synonym, you can change or delete the synonym.

Another reason you might want to modify the table is to limit the use of the ALL parameter for certain keywords by changing the parameter's default value from ALL=YES to ALL=NO or ALL=DIS. Using ALL=YES allows the operator to enter IMS commands with the ALL option; this requires a significant increase in storage in the IMS general pool and adversely affects IMS performance. To avoid these negative consequences, you can specify ALL=NO or ALL=DIS to be used with IMS commands except those associated with AOI transactions.

To obtain a listing of the command keyword table, print DFSCKWD0, a member of IMS.SDFSSRC. It contains the IMS keywords and synonyms described in *IMS Version 15.2 Commands, Volume 1: IMS Commands A-M*.

Restriction: DFSCKWD0 can only modify type-1 command keywords.

Details about ALL=NO and ALL=DIS options and instructions for modifying them are discussed in the following topics.

The following table shows the attributes of the IMS Command Language Modification facility.

Table 134. IMS command language modification facility attributes

Attribute	Description
IMS environments	DB/DC, DCCTL, DBCTL.
Naming convention	You must name this routine DFSCKWD0 with ALIAS CKWDTABL.

Binding	Example: This shows you how to bind the exit routine into IMS.SDFSRESL.
	`//LINKIT JOB 1,MSGLEVEL=1`
	`//LINK EXEC PGM=IEWL,PARM=(RENT,REFR,NCAL,XREF,LIST)`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20),RLSE)`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL,DISP=SHR`
	`//OBJIN DD DSN=IMS.USERLIB,DISP=SHR`
	`//SYSLIN DD *`
	`INCLUDE OBJIN(DFSCKWD0)`
	`MODE AMODE(31),RMODE(ANY)`
	`ALIAS  CKWDTABL`
	`ENTRY  DFSCKWD0`
	`NAME  DFSCKWD0(R)`
	`/*`

Including the routine	No special steps are needed to include this routine.
IMS callable services	IMS callable services are not applicable for use with this module.
Routine location	For the latest version of DFSCKWD0, see the IMS.SDFSSRC library; member name is DFSCKWD0.

Changing the table

Two of the macro statements that appear in the table, KEYWD and SYN, can be replaced to modify the keywords and synonyms. One way of modifying the table is:

1. Edit module DFSCKWD0.
2. Change the KEYWD and SYN macro statements.
3. Reassemble DFSCKWD0.
4. Relink the reassembled DFSCKWD0 in IMS.SDFSRESL.

Related Reading: *IMS Version 15.2 Commands, Volume 1: IMS Commands A-M* contains the list of reserved words, including command keywords, keyword synonyms, and reserved parameters.

Changes to DFSCKWD0 cannot conflict with the names in this list. Keywords can be changed and keyword synonyms can be added, changed, or deleted, as long as the new keyword or synonym is not a reserved word. For example, a new synonym of "MSDB" for MSDBLOAD cannot be added, because "MSDB" is a reserved parameter. If "MSDB" is made a keyword synonym, the `/DBDUMP DATABASE MSDB` command fails with a syntax error.

KEYWD macro statements must be substituted one-for-one in the table. No new KEYWD macro statements can be added.

KEYWD macro

KEYWD
keyword,LAST=NO|YES,ALL=YES|NO|DIS

Where *keyword* is the new or changed keyword. LAST=NO and ALL=YES are the defaults and need not be supplied. LAST=YES must be specified if it is the last macro call in the module. A keyword cannot exceed 12 characters in length.

Specifying ALL=NO prevents the use of the ALL parameter with all IMS commands that apply to the keyword being changed (except for commands issued from AOI programs).

For example, specifying ALL=NO for the keyword LTERM prevents the use of the ALL parameter for the following commands:

```
/BROADCAST LTERM ALL
/DISPLAY ASSIGNMENT LTERM ALL
/DISPLAY LTERM ALL
/LOCK LTERM ALL
/PSTOP LTERM ALL
/PURGE LTERM ALL
/START LTERM ALL
/STOP LTERM ALL
/UNLOCK LTERM ALL
```

Specifying ALL=DIS prevents the use of the ALL parameter with all /DISPLAY commands that apply to the keyword being changed (except for commands issued from AOI programs).

For example, specifying ALL=DIS for the keyword LTERM prevents the use of the ALL parameter for the following commands:

- `/DISPLAY ASSIGNMENT LTERM ALL`
- `/DISPLAY LTERM ALL`

SYN macro

SYN
synonym,LAST=YES|NO

Where *synonym* is the desired synonym. LAST=NO is the default and need not be specified. LAST=YES must be coded if this is the last macro call in the assembly. Synonyms cannot exceed 12 characters in length; they must be defined under the keyword to which they apply.

Error messages

Any error in a macro statement terminates assembly of the keyword table and results in generation of an error message. The remaining macro statements are error checked, but nothing is generated. All macro assembly errors are severity code 16 errors.

KYTBL001 - SEQUENCE ERROR. XXX CANNOT FOLLOW IKEY
A macro was called which cannot immediately follow an IKEY macro call. XXX is either IKEY or SYN. IKEY calls cannot be modified.

KYTBL002 - XXX CALLED WITHOUT ANY PARAMETER
A macro was called without any parameter. XXX is either IKEY, KEYWD, or SYN.

KYTBL003 - XXX IS NOT A VALID INTERNAL KEYWORD
The parameter specified on an IKEY call (XXX) is not known to the system. IKEY calls cannot be modified.

KYTBL004 - KEYWORD TABLE ASSEMBLY TERMINATED
This message appears as a comment after the first error message in a keyword table assembly. All subsequent macro calls will only perform error checking. No code will be generated.

KYTBL005 - SEQUENCE ERROR. KEYWD MUST FOLLOW AN IKEY CALL
A KEYWD macro was called that does not immediately follow an IKEY call.

KYTBL006 - LENGTH ERROR. XXX TOO LONG
The parameter specified on a KEYWD or SYN macro is more than 12 characters in length.

KYTBL007 - INTERNAL KEYWORD 'XXX' HAS NOT BEEN USED
LAST=YES was specified on either a KEYWD or SYN macro call, but not all internal keywords known to the system have been generated. IKEY calls cannot be modified. LAST=YES must appear only on the last macro call in the assembly.

KYTBL008 - XXX CANNOT BE SPECIFIED AGAIN
Internal keyword 'XXX' has already been used. IKEY macro calls cannot be modified.

KYTBL009 - KEYWD MACRO PARAMETER ALL IS INVALID
ALL=NO was erroneously specified on the KEYWD macro at the time the IMS Command Keyword Table (DFSCKWD0) was modified.

Message DFS058 COMMAND COMPLETED EXCEPT xxx y z... uses the keyword table to replace 'xxx' with the keyword associated with the command that caused the message. Therefore, keywords defined by KEYWD macro calls appear in this message. Other messages, however, are prebuilt, and keywords that might have changed will still appear in these.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample IMS Command Language Modification facility

The sample IMS Command Language Modification facility, located in the IMS.SDFSSRC library, demonstrates how to change the command keyword table.

For the latest version of DFSCKWD0, see the IMS.SDFSSRC library. The member name is DFSCKWD0.

IMS Initialization and Termination user exit

The IMS initialization and termination user exit is called early in the IMS initialization process and during normal and abnormal IMS termination. Use the IMS Initialization and Termination user exit to perform any setup that your user exits require. For example, use INITTERM during IMS initialization to allocate storage that is used to share information between user exits.

This topic contains Product-sensitive Programming Interface information.

About this routine

This exit is called only if it is defined in the DFSDFxxx member of the IMS.PROCLIB data set. There is no default exit routine for this exit. Multiple exit routines can be defined so that they are called sequentially.

Table 135. Initialization and termination exit routine attributes

Attribute	Description
IMS environments	All IMS regions
Naming convention	Any name can be used.

Table 135. Initialization and termination exit routine attributes (continued)

Attribute	Description
Including the routine	Specify EXITDEF=(TYPE=INITTERM,EXITS(*exit_names*)) in the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The exit routine module or modules must be included in an authorized library that is included in the JOBLIB, STEPLIB, or LINKLIST concatenation for the system.
IMS callable services	To use callable services with this exit routine, it must be given a callable services token by IMS at the time it is given control. Check the value of the SXPLATOK field in the "IMS standard user exit parameter list" on page 5: • If SXPLATOK is zero, you cannot use callable services with this routine. • If SXPLATOK is non-zero, the callable services token is included, and you can use callable services. You can use the 256-byte work area addressed by the SXPLAWRK field to call DFSCSIF0.
Sample routine location	IMS.SDFSSMPL (member name DFSITRX0)

Contents of the registers when the exit is called

Register
Contents

1
Address of the "IMS standard user exit parameter list" on page 5. This exit routine uses the Version 6 standard exit parameter list.

13
Save area address. The exit routine must not change the first three full words of this save area. This save area is not chained to any other save areas.

14
Return address.

15
Entry point of the exit routine.

Exit routine parameter list

The address for this parameter list is passed to the exit routine in the SXPLFSPL field of the "IMS standard user exit parameter list" on page 5. This parameter list is mapped by the DFSITXP macro.

Table 136. Parameter list for the Initialization and termination user exit type

Field name	Offset	Length	Usage	Description
ITXP_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001')
ITXP_FUNC	X'04'	X'04'	Input	Function code: **1** IMS initialization **2** IMS normal termination **3** IMS abnormal termination
ITXP_LEN	X'08'	X'04'	Input	Parameter list length

Table 136. Parameter list for the Initialization and termination user exit type (continued)

Field name	Offset	Length	Usage	Description
ITXP_RGNTYPE	X'0C'	X'04'	Input	Region type: **1** DB/DC **2** DBCTL **3** DCCTL **4** FDBR

Contents of the registers when the routine returns control to IMS

There is no requirement for exit registers and there are no defined return and reason codes.

IMS Monitor user exit (IMSMON)

The IMS Monitor user exit provides access to the IMS Monitor data without the need to modify IMS code. The exit is called during IMS initialization, IMS termination, and anytime a Monitor log record is written, regardless whether the DC Monitor is on or off.

This topic contains Product-sensitive Programming Interface information.

About this routine

This optional exit is called by IMS only if you define it to IMS on the EXITDEF parameter with TYPE=IMSMON in the USER_EXITS section of the DFSDF*xxx* member. There is no default exit name.

When the FASTMON and the IMSMON user exits are both defined, the FASTMON user exits are called before the IMSMON user exits in the order that the exits are listed in the EXITDEF parameter. After all the FASTMON exits are called, the IMSMON exits are called in the same order.

IMSMON exit routines must register interest in the SLOG codes (field name SL_CODE) that they are to be called for. However, because multiple user exit routines might be defined for the IMSMON user exit, and other routines could register interest in more SLOG codes, a user exit routine might be driven for SLOG codes for which it does not register interest. The exit routines perform this registration when called with the INIT function code.

This exit is also called after a successful **REFRESH USEREXIT** command with an **INIT** call function code. This behavior allows the users to reformat storage that was allocated or to do any other tasks that must be done to allow new versions of other exit routines to be called. The IMSMON user exit type also allows the exit routines for this type to change which SLOG codes the routines are interested in. If the IMSMON exit is successfully refreshed, the new copy of the exit routine is called.

If the exit type is deleted with the **REFRESH USEREXIT** command from a DFSDF*xxx* member that does not have any IMSMON exits listed, then a TERM call is made to the IMSMON exit routine or routines before they are deleted.

Under rare circumstances, the IMSMON exit would not be called and either a DFS4573E or DFS4587E message would be issued. These circumstances are as follows:

- Not being able to obtain an AWE for a BCB FUNC=GET call.
- Not being able to obtain a BCB block for the interface block and parameter list for the exit.
- Not being able to obtain the storage for the SLOG interest array that is used by user exit services to check whether at least one exit is interested in a particular SLOG code.

All these circumstances are storage shortage issues where small amounts of storage (a few hundred bytes at most) cannot be obtained.

Table 137. IMS Monitor exit routine attributes

Attribute	Description
IMS environments	All IMS regions.
Naming convention	Any name can be used. Refer to "Exit routine naming conventions" on page 3 for guidelines on routine names.
Binding	Follow the guidelines in "Routine binding restrictions" on page 9.
Including the routine	You must specify EXITDEF=(TYPE=IMSMON,EXITS=(*exit_names*)) in the EXITDEF parameter in the USER_EXITS section of the DFSDFxxx member. The module must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
IMS callable services	This exit is eligible to use IMS callable services. The callable services token is passed in the standard user exit parameter list (SXPL).
Sample routine location	IMS.SDFSSMPL (member name DFSMONX0).

Contents of the registers when the exit is called

Register
Contents

1
Address of SXPL.

13
Save area address. The exit must not change the first three words of this save area. This save area is not chained to any other IMS save areas.

14
Return address to IMS.

15
Entry point of exit routine.

The WRITE call function for the IMSMON exit can run in any address space that is associated with an IMS control region (IMS control, DLI/SAS, or dependent region). The exit might be given control in cross-memory mode to either control or DLI.

Exit routine parameter list

This exit routine uses the current version of the standard exit parameter list.

The IMS Monitor exit routine parameter list is mapped by DSECT MONEXPL in macro DFSLOGP.

Table 138. Parameter list for the IMS Monitor user exit type

Field name	Offset	Length	Usage	Description
MONEXTYP	X'00'	X'01'	Input	Function code in decimal: **1** IMS initialization. **2** Write call from DFSSLOG. **3** Termination.

Table 138. Parameter list for the IMS Monitor user exit type (continued)

Field name	Offset	Length	Usage	Description
	X’01’	X’03’	None	Reserved.
MONEXVRS	X’04’	X’04’	Input	Parameter list version. The version number is 3 after applying APAR PH24963.
MONEXECB	X’08’	X’04’	Input	The ECB passed to the Monitor from the DFSSLOG macro invocation. This ECB might be different than the ECB that represents the currently dispatched ITASK that called the Monitor. This field is 0 during INIT and TERM calls because DFSSLOG does not originate those calls. This ECB might have an outstanding POST active, or it might contain a post code that the DFSSLOG invoker might depend on. This ECB should not be modified by the IMSMON user exit or used on system service calls. See MONEXXEC.
MONEXPRM	X’0C’	X’04’	Input	Address of the DFSSLOG parameter list that is mapped by DSECT SLOGPRM in macro DFSLOGP.
MONEXTBL	X’10’	X’04’	Output	Used only in initialization calls (normal initialization and the INIT call after REFRESH). Address of the array where the exit registers interest in certain SLOG codes. Note that this field is set to x'FFFFFFFF' if IMS cannot obtain storage for the table. In this case, the IMSMON exit is called for every monitor SLOG code. This field is 0 for WRTE and TERM calls.
MONEXCST	X'14'	X'08'	Input	CSECT name of the DFSSLOG invoker. This field is not populated for INIT and TERM calls because they do not come from DFSSLOG.
MONEXXEC	X'1C'	X’04’	Input	Address of the currently dispatched ECB when the exit routine is called. This ECB is used for things such as system service calls. This field is present only if MONEXVRS is MONEXV2 or greater. This field is 0 for INIT and TERM calls.
MONEXFPT	X'20'	X'04'	Input	Pointer to a byte storage. This field is defined only when MONEXVRS is 3 or greater. It only applies to the FASTMON user exit type and is zero for IMSMON user exits.

Registering Interest in SLOG codes

When the exit is called during monitor initialization (function code 1), the exit must register which SLOG codes it would like to be called for. The exit registers the interest by moving the character 'Y' into the array that it is passed (field name MONEXTBL) into the offset that corresponds to the SLOG code that it is

interested in. For example, if the exit is to register interest in SLOG code x'50', then it moves the character 'Y' into the location that is pointed to by MONEXTBL + x'50'. If IMS cannot get storage for the array to pass to the exit, the field MONEXTBL contains x'FFFFFFFF', and the exit does not need to register interest in any SLOG codes because the exit is called for all SLOG codes.

Refer to the sample exit in IMS.SDFSSMPL(DFSMONX0) for an example of how to register interest in SLOG codes for the INIT call.

An exit must not modify any other places in the array other than moving a 'Y' into the slot that represents the SLOG codes that it is interested in. Because all exit routines for the IMSMON exit type are passed the same array, moving something other than a 'Y' into a slot would unregister another exit's interest in that SLOG code if that exit also puts a 'Y' there.

The FASTMON exit and the IMSMON exit share one interest array. If one of the exits registers interest for an SLOG code, both exits are called.

When both FASTMON and IMSMON user exits are defined, IMSMON exits can add subcodes to the interest array only when the exits are refreshed. Subcodes cannot be removed from the array.

Related reference

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

IMS Fast Monitor User Exit (FASTMON)

The IMS Fast Monitor User Exit (FASTMON) provides access to the IMS Monitor data. The FASTMON exit is called during IMS initialization, IMS termination, and at IMS monitor call points for registered SLOG codes, regardless of whether the IMS Monitor is on or off.

The IMS Fast Monitor (FASTMON) user exit provides a similar exit point as the IMSMON user exit does. However, the FASTMON exit is optimized to minimize performance usage, in exchange for certain functional restrictions. The FASTMON user exit:

- Not refreshable;
- Cannot use user exit callable services;
- Has a simple register interface for the WRITE call.

All defined FASTMON user exits are called whenever the IMS monitor module DFSMNTR0 is called for a monitor event when any FASTMON or IMSMON user exit has registered interest for the specific SLOG code.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

Defining the routine

The FASTMON user exit is specified by an EXITDEF statement with TYPE=FASTMON in the DFSDFxxx USER_EXITS section. You can specify a single exit module name or a list of module names for this user exit. The exits are loaded at IMS control region initialization and persist until IMS termination.

The FASTMON exits are not refreshable.

When the FASTMON and the IMSMON user exits are both defined, the FASTMON user exits are called before the IMSMON user exits in the order that the exits are listed in the EXITDEF parameter. After all the FASTMON exits are called, the IMSMON exits are called in the same order.

Calling the routine

Each FASTMON user exit is called for initialization (function code 1), write (function code 2), and termination (function code 3) functions. When the FASTMON exit is called for initialization or termination functions, it is passed with the address of a monitor exit parameter list (MONEXPL), which is defined in macro DFSLOGP. When the FASTMON exit is called for a write function, the SLOG parameter list that was passed to IMS monitor is passed to the FASTMON exit.

A FASTMON user exit can be dynamically enabled or disabled. When it is called for initialization, MONEXFPT in DSECT MONEXPL points to a byte in storage which is used to enable or disable the exit. Refer to section Enabling and disabling exit for WRITE function calls below for further details.

Circumstances where the routine is not called

Under rare circumstances, the FASTMON exit is not called and one of DFS4573E, DFS4588E, or DFS4570E message is issued:

- IMS is unable to obtain an AWE for a BCB FUNC=GET call.
- IMS is unable to obtain a BCB block for the parameter list for the exit.
- IMS is unable to obtain the storage for the SLOG interest array that is used by user exit services to check whether at least one exit is interested in a particular SLOG code.
- IMS is unable to obtain the storage for setting up the FASTMON exit list.

All these circumstances are storage shortage issues where small amounts of storage (a few hundred bytes at most) cannot be obtained.

Table 139. IMS Fast Monitor exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	Any name can be used. Refer to “Exit routine naming conventions” on page 3 for guidelines on routine names.
Binding	Follow the guidelines in “Routine binding restrictions” on page 9.
Including the routine	You must specify EXITDEF=(TYPE=FASTMON, EXITS=(exit_names)) in the EXITDEF parameter in the USER_EXITS section of the DFSDFxxx member. The module must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
IMS callable services	This exit does not support IMS callable services.
Sample routine location	IMS.SDFSSMPL (member name DFSFMON0)

Contents of the registers when the exit is called

The contents of the registers vary depending on the function the FASTMON exit is called for:

When a FASTMON exit is called for an INIT or TERM function:

Register
Contents

R0
A(SCD)

R1
A(MONEXPL) (defined in the DFSLOGP macro)

R13
A(save area) in an IMS save area set. The exit might save its caller's registers in this save area set, but must not modify the save area's forward and backward chain pointers.

R14
Return address

R15
Exit entry point address

When a FASTMON exit is called for a WRITE function:

Register
Contents

R0
A(SCD)

R1
A(SLOG parmlist) (DSECT SLOGPRM, defined in the DFSLOGP macro) with the low X'00000001' bit set as a flag to tell the exit that it is the WRITE call

R13
A(save area) in an IMS save area set. The exit might save its caller's registers in this save area set, but must not modify the save area's forward and backward chain pointers.

R14
Return address

R15
Exit entry point address

Note: The SCD and the SLOG parameter lists are classified as DMTI (diagnosis, modification, and tuning information). Any fields in the SCD or the SLOG parameter lists are subject to change, addition, or deletion at any time (both between IMS releases and via maintenance).

The WRITE call function for the FASTMON exit can run in any address space that is associatedthat is with an IMS control region (IMS control, DLI/SAS, or dependent region). The exit might be given control in cross-memory mode to either control or DLI. The caller's ITASK ECB might have an outstanding event on it and must not be modified or used for any WAIT/POST type processing.

Differentiating between an INIT/TERM call and a WRITE call

The FASTMON user exit can differentiate between an INIT/TERM call and a WRITE call by using the low bit of register 1:

INIT/TERM
The low bit is zero and R1 points to a MONEXPL.

WRITE
The low bit is one, and R1 points to an SLOG parameter list.

Note that when the low bit of R1 is set (WRITE call), the user exit must clear the low bit before using the value in R1 to address the SLOG parameter list. The following is an example of how to check the R1 value and clear the flag:

```
           TML   R1,X'0001'  Is this a WRITE call?
           JNO   INITTERM    No, init/term; R1 -> MONEXPL
           NILL  R1,X'FFFE'  Yes, clear flag; R1 -> DFSSLOG
*
* Process WRITE call - R1 = A(DFSSLOG)
*
           USING SLOGPRM,R1  Address SLOG parmlist
           . . .
           J     ENDPROC     Done with processing
*
* Process INIT/TERM call - R1 = A(MONEXPL)
*
INITTERM DS    0H
           USING MONEXPL,R1  Address monitor exit parmlist
```

```
          . . .
ENDPROC  DS    0H
```

Exit routine parameter list

The IMS Fast Monitor exit routine parameter list is mapped by DSECT MONEXPL in macro DFSLOGP.

Table 140. Parameter list for the IMS Fast Monitor user exit types

Field name	Offset	Length	Usage	Description
MONEXTYP	X'00'	X'01'	Input	Function code in decimal: **1** IMS initialization. **3** Termination.
MONEXFL1	X'01'	X'01'	Input	Flag byte 1. This field is only defined when MONEXVRS is 3 or greater. Bit definitions: X'80' (MONEXFST) - Monitor exit that is called is FASTMON exit.
	X'02'	X'02'	None	Reserved.
MONEXVRS	X'04'	X'04'	Input	Parameter list version. The current version number is 3.
MONEXECB	X'08'	X'04'	Input	Not applicable to FASTMON; the value is 0.
MONEXPRM	X'0C'	X'04'	Input	Not applicable to FASTMON; the value is 0.
MONEXTBL	X'10'	X'04'	Output	Used only in initialization calls. Address of the array where the exit registers interest in certain SLOG codes. Note that this field is set to x'FFFFFFFF' if IMS cannot obtain storage for the table. In this case, the FASTMON exit will be called for every monitor SLOG code. This field is 0 for WRITE and TERM calls.
MONEXCST	X'14'	X'08'	Input	Not applicable to FASTMON; the value is 0.
MONEXXEC	X'1C'	X'04'	Input	Not applicable to FASTMON; the value is 0.
MONEXFPT	X'20'	X'04'	Input	Pointer to a byte storage (see 'Enabling and disabling exit for WRITE function calls' section). This field is only defined when MONEXVRS is 3 or greater and applies only to the FASTMON user exit type.

Registering Interest in SLOG codes

When the exit is called during monitor initialization (function code 1), the exit must register which SLOG codes it would like to be called for. The exit registers the interest by moving the character 'Y' in an array (field name MONEXTBL). The array is passed to the offset that corresponds to the SLOG code that the exit is interested in. For example, if the exit wants to register an interest in SLOG code x'50', it moves the character 'Y' to the location that `MONEXTBL + x'50'` points to. If IMS cannot get storage for the array to pass to the exit, the field MONEXTBL contains x'FFFFFFFF'. The exit does not need to register interest in any SLOG codes because the exit is called for all SLOG codes.

Refer to the sample exit in IMS.SDFSSMPL(DFSFMON0) for an example of how to register interest in SLOG codes for the INIT call.

An exit must not modify any other places in the array other than moving a 'Y' in the slot that represents the SLOG codes that it is interested in. Because all exit routines for the IMSMON and FASTMON exit types are passed in the same array, moving something instead of a 'Y' in a slot would unregister another exit's interest in that SLOG code if that exit also puts a 'Y' there.

The FASTMON exit and the IMSMON exit share one interest array. If one of the exits registers interest for an SLOG code, both exits get called.

The FASTMON exit can only add subcodes to the interest array.

Enabling and disabling exit for WRITE function calls

DSECT MONEXPL provides a pointer to a byte used to enable and disable the exit for the subsequent WRITE function calls.

MONEXFPT is defined in DSECT MONEXPL and is a pointer to a byte in key 7 common storage that can be used to enable and disable monitor write calls (MONEXWRT) to the FASTMON user exit. If a user exit wants to dynamically enable and disable calls to itself, it can save the MONEXFPT pointer that is passed from the initialization call and update the byte pointed to it to control calls for write.

Valid values for this byte are:

MONEXWRE
Write calls enabled

MONEXWRD
Write calls disabled

IMS initializes the value of the byte to MONEXWRE, so no use of this pointer is necessary if the exit does not support dynamic enable or disable.

When you enable or disable the FASTMON exit for WRITE function calls, be aware that:

1. Always check whether MONEXVRS is equal to or greater than MONEXV3 before you reference this field – This field is only present when the parameter list version (MONEXVRS) is 3 (MONEXV3) or higher.
2. Always check whether the pointer is 0 before using it to access the byte – This pointer is only passed on the initialization call (MONEXINT) and only for the FASTMON exit type.

   Attention: There is no dynamic enable or disable for the IMSMON exit. The value is 0 for the IMSMON exit type.
3. The byte is only used to control calls to the exit for monitor write (MONEXWRT). The termination call (MONEXTRM) is made regardless of the byte's setting.
4. The storage that contains the control byte is freed when IMS terminates. If you are updating the byte asynchronously to your FASTMON exit execution, you must ensure that IMS is still active. Failure to do so can result in storage overlays.
5. If you set this byte to anything instead of MONEXWRE or MONEXWRD, the monitor exit remains enabled. However, future changes might alter this behavior:
   - Set the byte that is pointed to by MONEXFPT or MONEXWRD to disable monitor write calls to the FASTMON user exit.

Note: Because the monitor might be called in parallel under many different IMS transaction control blocks (TCBs), the user exit might still be called for a short period of time after the byte is set to MONEXWRD due to timing conditions.

- Set the byte that is pointed to by MONEXFPT or MONEXWRE to enable monitor write calls to the FASTMON user exit.

  Note: When calls to the user exit are currently disabled, this must be done by a process outside of the user exit itself.

Language Environment User exit routine (DFSBXITA)

Use the Language Environment User exit routine to dynamically update IBM Language Environment for z/OS runtime parameters for an IMS application.

This topic contains Product-sensitive Programming Interface information.

This topic describes the CEEBXITA Assembler User exit routine, DFSBXITA, and provides information about the attributes of the routine, how the routine is called and how the routine communicates with IMS.

Subsections:

- "About this routine" on page 391
- "Communicating with IMS" on page 392

About this routine

DFSBXITA is the IMS-specific version of the LE-defined user exit CEEBXITA. To use dynamic runtime parameters with an IMS application, DFSBXITA must be linked as CEEBXITA in one of two ways:

- Linked with the application, in which case DFSBXITA functions as an application-specific user exit
- Linked with the LE initialization/termination library routines, in which case DFSBXITA functions as an installation-wide user exit

Note: If your z/OS environment includes software other than IMS that uses CEEBXITA, be aware that if DFSBXITA is linked with the LE initialization/termination library routines, it will be called by the non-IMS software that previously called CEEBXITA. You must provide logic to ensure that programs that need to use CEEBXITA can access it.

DFSBXITA executes only when the first routine in an LE enclave is initialized. It ignores all other calls. DFSBXITA issues a DL/I INQY LERUNOPT call to determine if applicable runtime override parameters exist. If so, the INQY call returns the address of the parameter string. DFSBXITA returns that address to LE in the field CEEAUE_OPTION. The string includes a halfword length field followed by the dynamic runtime parameters as they are specified to IMS. The length of the string does not include the length field. A zero is returned in CEEAUE_OPTION if:

- No dynamic runtime override parameters exist.
- An error occurs during exit processing.
- An error occurs during INQY call processing.

Table 141. Language Environment user exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL and DCCTL.
Naming convention	Must be named CEEBXITA.
Binding	After you compile your routine, include it into IMS.SDFSRESL or into any operating system-partitioned data set to which access is provided by using a JOBLIB or STEPLIB JCL statement.

Table 141. Language Environment user exit routine attributes (continued)

Attribute	Description
Including the routine	No special steps required.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	IMS.ADFSSMPS

Calling this routine

LE calls this exit routine using standard linkage conventions.

Communicating with IMS

IMS communicates with this routine through the entry registers, a parameter list, and the exit registers.

Content of Registers on Entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Input parameter list address (CEEAUE)
12	Pointer to the common anchor area (CAA), which is mapped by CEECAA
13	Caller's save area address
14	Return address
15	Entry point address

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Log Archive exit routine

The Log Archive exit routine produces an edited subset of the complete IMS log.

Subsections:

About this routine

You can use the sample Log Archive exit routine to produce an edited subset of the complete IMS log. The subset log contains the records needed by the Tivoli® Performance Reporter z/OS, (Program Number 5695-101). The Tivoli Performance Reporter z/OS (PR) collects statistics about IMS transactions and schedules.

Restriction: The IBM-supplied sample exit routine is applicable only to an IMS DB/TM system and should not be used in a DBCTL environment.

However, a user-written exit routine can run in a DBCTL environment.

The following table shows the attributes of the Log Archive exit routine.

Table 142. Log archive exit routine attributes

Attribute	Description
IMS environments	Only used by the Log Archive utility.
Naming convention	Must match name specified on Log Archive EXIT statement.

Binding	You must bind the exit routine into RESLIB (or a library concatenated with it) as a separate **reentrant** or **reusable** load module. If the module is not present in the load library, the IMS Log Archive utility does not load or call it.
	Example: This shows you how to bind the exit routine into IMS.SDFSRESL.
	`//LINKIT JOB 1,MSGLEVEL=1`
	`//LINK EXEC PGM=IEWL,PARM=RENT`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL.,DISP=SHR`
	`//OBJIN DD DSN=IMS.USERLIB.,DISP=SHR`
	`//SYSLIN DD *`
	`INCLUDE OBJIN(IMSEXIT)`
	`MODE AMODE(24),RMODE(24)`
	`NAME IMSEXIT(R)`
	`/*`

Including the routine	Use the Utility Control EXIT statement.
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	No sample exit routine is provided.

Attributes of the routine

You must write this exit routine in assembler language. This exit routine receives control running in 24-bit addressing mode and must return control in that mode.

Using IMS callable services with this routine

This exit routine is not eligible to use IMS callable services.

Restrictions

An abend in the exit routine causes the utility to abend. IMS macros cannot be used in the exit routine. Because the performance of the exit routine affects the total performance of the utility, the logic of the exit routine should not be so complicated as to delay the OLDS from being used by the online region.

Communicating with IMS

IMS communicates with the Log Archive exit routine through the entry registers, parameter list, and exit registers.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to the calling RECON access routine.
15	Entry point of exit routine.

Parameter list

The parameter list contains the following:

Exit Routine Word (Word 1)
This word belongs to the exit routine. On the initialization call entry to the exit routine, this word contains binary 0. The routine can store any value in this word. For example, the word can point to an area allocated for use by the exit routine through the GETMAIN macro. On subsequent calls to the exit routine, this field contains the value left by the routine on its previous invocation.

Call Type Indicator Field Address (Word 2)
Address of a one-byte area containing the call type indicator.

X'01'
Initialization call

X'02'
Log record processing call

X'03'
Termination call

The call type indicator identifies the reason for calling the exit routine, and the exit routine can have a separate routine for each call type. The user exit should not change this field.

Address of Area Containing the Current Input Log Record or Utility Return code (Word 3)
The high-order bit of this word is ON to indicate the end of the list. The contents of the remainder of the word depends on the type of call:

- On an initialization call, this word is zero.
- On a log record processing call, this word will have the address of an area containing the current input log record. The first four bytes of the log record are a BSAM RDW (Record Descriptor Word).
- On a termination call, this word will contain the return code for the current utility.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

X'00'
Active utility continues processing

non-0
Active utility terminates and IMS issues an error message

Termination due to the exit routine on an initialization call or a log record processing call prevents successful execution of the utility. Termination due to the routine on a termination call results in an error message and a nonzero return code, but successful execution is not prevented, because DBRC has already been notified of archive completion.

Related reference

“Routine binding restrictions” on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample Log Archive exit routine

Use the Sample Log Archive exit routine to review the record types, DCB information, and truncation instances in an edited subset of the complete IMS log.

The following sample shows the Sample Log Archive exit routine.

```
IMSEXIT  CSECT ,
**START OF MODULE SPECIFICATION*****************************************
*                                                                       *
*  MODULE-NAME = IMSEXIT                                                *
*                                                                       *
*  DESCRIPTIVE-NAME = SAMPLE IMS ARCHIVE FUNCTION EXIT                  *
```

```
*                                                                     *
*  COPYRIGHT = NONE                                                   *
*                                                                     *
*                                                                     *
*  FUNCTION:                                                          *
*    WRITES THE RECORDS USED BY SLR V3 (IBM PP, PROG NO 5665-397)     *
*    INTO THE FILE WITH DDNAME IMSLOG. THE FOLLOWING RECORD TYPES     *
*    ARE WRITTEN (ALL IN HEX): 01, 03, 06, 07, 31, 34, 35, 36, 38,    *
*    4001, 4003, 4004, 4098, 42. MESSAGE TEXTS OF 01 AND 03           *
*    RECORDS ARE TRUNCATED TO 24 BYTES.                               *
*                                                                     *
*  LOGIC:                                                             *
*    CASE INIT.                                                       *
*      GETMAIN STORAGE FOR WORK AREAS AND ANCHOR IT IN THE USER       *
*      WORD.                                                          *
*      OPEN OUTPUT FILE.                                              *
*    END CASE INIT.                                                   *
*                                                                     *
*    CASE NORMAL.                                                     *
*      SUBCASE RECORD TYPES 01, 03.                                   *
*        CALCULATE TOTAL LENGTH OF ALL TEXT SEGMENTS.                 *
*        IF (LENGTH OF ALL TEXT SEGMENTS > 24 BYTES) THEN             *
*          TRUNCATE ANY MESSAGE PART TO 24 BYTES.                     *
*          CHANGE SIGN OF TOTAL TEXT LENGTH AND STORE IT BACK AS AN   *
*          INDICATOR.                                                 *
*        ELSE.                                                        *
*                                                                     *
*        COPY RECORD.                                                 *
*      END SUBCASE RECORD TYPES 01, 03.                               *
*                                                                     *
*      SUBCASE RECORD TYPES 06, 07, 31, 34, 36, 38, 42.               *
*        COPY RECORD.                                                 *
*      END SUBCASE RECORD TYPES 06, 07, 31, 34, 36, 38, 42.           *
*                                                                     *
*      SUBCASE RECORD TYPES 4001, 4003, 4004, 4098.                   *
*        COPY RECORD.                                                 *
*      END SUBCASE RECORD TYPES 4001, 4003, 4004, 4098.               *
*    END CASE NORMAL.                                                 *
*                                                                     *
*    CASE TERMINATE.                                                  *
*      CLOSE OUTPUT FILE.                                             *
*      FREEMAIN STORAGE FOR WORK AREAS AND RESET ANCHOR POINTER.      *
*    END CASE TERMINATE.                                              *
*                                                                     *
*  NOTES = SEE BELOW                                                  *
*                                                                     *
*    DEPENDENCIES = NONE                                              *
*                                                                     *
*    RESTRICTIONS = NONE                                              *
*                                                                     *
*    REGISTER CONVENTIONS = SEE LINKAGE                               *
*                                                                     *
*    PATCH LABEL = NONE                                               *
*                                                                     *
*  MODULE-TYPE = PROCEDURE                                            *
*                                                                     *
*    PROCESSOR = ASSEMBLER                                            *
*                                                                     *
*    MODULE-SIZE = SEE ASSEMBLER LISTING                              *
*                                                                     *
*    ATTRIBUTES = REENTRANT                                           *
*                                                                     *
*  ENTRY-POINT = IMSEXIT                                              *
*                                                                     *
*    PURPOSE = SEE FUNCTION                                           *
*                                                                     *
*    LINKAGE = STANDARD OS LINKAGE                                    *
*                                                                     *
*  INPUT:                                                             *
*    REGISTER 1 POINTS TO A 3-WORD PARAMETER LIST:                    *
*                                                                     *
*    USERWORD - PTR(31). CONTAINS ZERO AT INIT CALL, AND A POINTER    *
*               TO A WORKAREA AT NORMAL AND TERM CALLS.               *
*    TYPEPTR  - PTR(31). POINTS TO A 1-BYTE AREA, THAT CONTAINS:      *
*               X'01' - INIT CALL                                     *
*               X'02' - NORMAL CALL                                   *
*               X'03' - TERM CALL                                     *
*    RECPTR   - PTR(31). FOR NORMAL CALL, POINTER TO A LOG RECORD.    *
*                                                                     *
*  FEEDBACK:                                                          *
*    USERWORD - PTR(31). FILLED IN WITH A POINTER TO A GETMAINED      *
*               WORK AREA AT INIT CALL.                               *
```

```
*                                                                     *
*  OUTPUT:                                                            *
*    SELECTED LOG RECORDS WRITTEN TO DDNAME IMSLOG                    *
*                                                                     *
*  MESSAGES:                                                          *
*    001 - UNABLE TO GET STORAGE                                      *
*    002 - UNABLE TO OPEN FILE IMSLOG                                 *
*    003 - ERROR DURING PUT TO IMSLOG                                 *
*    004 - INVALID CALL TYPE                                          *
*                                                                     *
*  ABEND CODES:                                                       *
*    NONE.                                                            *
*                                                                     *
*  EXTERNAL-REFERENCES = NONE                                         *
*                                                                     *
*  ASSEMBLER MACROS:                                                  *
*    DCB                                                              *
*    DCBD                                                             *
*    FREEMAIN                                                         *
*    CLOSE                                                            *
*    GETMAIN                                                          *
*    OPEN                                                             *
*    PUT                                                              *
*                                                                     *
*                                                                     *
* NOTES:                                                              *
*   THE FOLLOWING REGISTERS ARE IN THE CODE:                          *
*                                                                     *
*     R6  = RECPTR:  POINTER TO THE INPUT RECORD                      *
*     R9  = PBLDREC: POINTER TO THE RECORD TO WRITE                   *
*     R10 = ENTIND:  ENTRY TYPE                                       *
*                                                                     *
**END OF MODULE SPECIFICATION*****************************************
*---------------------------------------------------------------------*
*                                                                     *
*      PROLOG CODE                                                    *
*      - SET UP ADDRESSABILITY                                        *
*      - GETMAIN STORAGE IF INIT CALL                                 *
*                                                                     *
*---------------------------------------------------------------------*
         USING *,R15
         B     PROLOG              * BRANCH PAST MODULE ID
         DC    AL1(16)             * MODULE ID LENGTH
         DC    C'IMSEXIT   82.103'  * MODULE ID
         DROP  R15
PROLOG   STM   R14,R12,12(R13)      * SAVE REGS
         LR    R12,R15             * SET NEW BASE REG
         USING IMSEXIT,R12         * SET ADDRESSABILITY
         LR    R11,R1              * SAVE PARM LIST ADDRESS
         L     R1,0(,R1)           * GET WORK AREA POINTER (OR 0)
         L     R7,4(,R11)          * COPY
         SLR   ENTIND,ENTIND       *   ENTRY
         IC    ENTIND,0(,R7)       *     INDICATOR
         LTR   ENTIND,ENTIND       * IS ENTRY TYPE ZERO ?
         BZ    OTHCASE             * YES, SKIP TO ISSUE MSG
         CL    ENTIND,TERMCALL     * IS ENTRY TYPE TOO GREAT ?
         BH    OTHCASE             * YES, SKIP TO ISSUE MSG
         C     ENTIND,INITCALL     * NO, IS THIS INIT ENTRY ?
         BNE   NOGETMAN            * NO, DON'T ISSUE GETMAIN
         L     R0,SIZEWORK         * GET SIZE OF DYNAMIC AREA
         GETMAIN  R,LV=(0)         * GET STORAGE FOR DYNAMIC AREA
         LTR   R15,R15             * REQUEST OK ?
         BZ    GETMOK              * YES, SKIP ON
*                                  * NO, SEND A MESSAGE
         WTO   'IMSE001 - UNABLE TO GET STORAGE',ROUTCDE=11,DESC=7
         B     NOFREEMN            * SKIP TO EPILOG & RETURN
GETMOK   ST    R1,0(,R11)          * SAVE ADDR IN USER WORD
NOGETMAN LTR   R1,R1               * ANY STORAGE GOTTEN ?
         BZ    NOFREEMN            * NO, SKIP TO EPILOG & RETURN
         USING WORKAREA,R1         * SET TEMP LOCATE OF NEW SAVEAREA
         ST    R13,SAVEAREA+4      * SET CHAIN BACK PTR IN NEW SAVEAR
         DROP  R1                  * DROP TEMP LOCATE
         ST    R1,8(,R13)          * SET CHAIN FORWARD PTR IN OLD SAV
         LR    R13,R1              * POINT TO NEW SAVE AREA
         USING WORKAREA,R13        * LOCATE NEW SAVEAREA
         L     RECPTR,8(,R11)      * COPY RECORD POINTER
         SLR   PBLDREC,PBLDREC     * ZERO OUTPUT RECORD POINTER
         C     ENTIND,INITCALL     * IS THIS INITIAL CALL ?
         BNE   NOTINIT             * IF NOT, SKIP ON
*---------------------------------------------------------------------*
*                                                                     *
*      INIT CALL                                                      *
```

```
*                                                                     *
*---------------------------------------------------------------------*
         MVC   DYNDCB(LENDCB),LISTDCB * COPY STATIC TO DYNAMIC DCB
         OI    DYNOPEN,X'80'          * SET HIGH ORDER BIT IN OPEN LIST
         LA    R5,DYNDCB              * POINT TO DYNAMIC DCB
         OPEN  ((R5),OUTPUT),MF=(E,DYNOPEN) * OPEN DYNAMIC DCB
         USING IHADCB,R5              * LOCATE DCB
         TM    DCBOFLGS,DCBOFOPN      * OPEN OK ?
         BO    EPILOG                 * YES, SKIP TO EPILOG
*                                     * NO, SEND A MESSAGE
         WTO   'IMSE002 - UNABLE TO OPEN FILE IMSLOG',ROUTCDE=11,DESC=7
         L     ENTIND,TERMCALL        * INDICATE TO TERMINATE
         B     EPILOG                 * SKIP TO EPILOG AND RETURN
         DROP  R5                     * DROP DCB ADDRESS
*---------------------------------------------------------------------*
*                                                                     *
*      NORMAL CALL                                                    *
*                                                                     *
*---------------------------------------------------------------------*
NOTINIT  C     ENTIND,NORMCALL        * IS THIS NORMAL CALL ?
         BNE   TERMCASE               * IF NOT, SKIP TO TERMINATE CASE
         SLR   R7,R7                  * INSERT RECORD TYPE
         IC    R7,RECTYPE(,RECPTR)    * INTO WORK REGISTER
*---------------------------------------------------------------------*
*      BRANCH TO APPROPRIATE RECORD PROCESSING ROUTINE                *
*      VIA BRANCH TABLE                                               *
*---------------------------------------------------------------------*
         C     R7,TYPE01              * RECORD TYPE 01 ?
         BL    RECEND                 * NO, SOMETHING LESS, IGNORE IT
         BE    REC0103                * YES, GO PROCESS IT
         C     R7,TYPE42              * NO, RECORD TYPE 42 ?
         BH    RECEND                 *   NO, SOMETHING LARGER, IGNORE
         BE    RECCOPY                *   YES, GO PROCESS IT
         BCTR  R7,0                   *   CONVERT RECORD TYPE 02 - 41
         SLL   R7,2                   *     TO A 4-BYTE INDEX
BRANCHTB B     BRANCHTB(R7)           *       USED TO BRANCH IN TABLE
         B     RECEND                 * RECORD TYPE 02, NOT USED HERE
         B     REC0103                * RECORD TYPE 03
         DC    2S(X'7F0'(4),RECEND) * RECORD TYPES 04 AND 05, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 2 BRANCHES TO RECEND
         B     RECCOPY                * RECORD TYPE 06
         B     RECCOPY                * RECORD TYPE 07
         DC    41S(X'7F0'(4),RECEND) * RECORD TYPES 08 - 30, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 41 BRANCHES TO RECEND
         B     RECCOPY                * RECORD TYPE 31
         DC    2S(X'7F0'(4),RECEND) * RECORD TYPES 32 AND 33, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 2 BRANCHES TO RECEND
         B     RECCOPY                * RECORD TYPE 34
         B     RECCOPY                * RECORD TYPE 35
         B     RECCOPY                * RECORD TYPE 36
         B     RECEND                 * RECORD TYPE 37, NOT USED HERE
         B     RECCOPY                * RECORD TYPE 38
         DC    7S(X'7F0'(4),RECEND) * RECORD TYPES 39 - 3F, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 7 BRANCHES TO RECEND
         B     REC40                  * RECORD TYPE 40
         B     RECEND                 * RECORD TYPE 41, NOT USED HERE
*---------------------------------------------------------------------*
*      RECORD PROCESSOR FOR 01, 03 RECORDS                            *
*      - COPY AT MOST 24 BYTES OF MESSAGE TEXT                        *
*---------------------------------------------------------------------*
REC0103  DS    0H                     * PROCESS RECORDS 01, 03
         LA    PBLDREC,RECAREA        * POINT TO OUTPUT BUFFER
         LH    R2,RECPRELL(,RECPTR) * LOAD RECORD PREFIX LENGTH
         LH    R7,RECLL(R2,RECPTR)  * SAVE LENGTH OF 1ST SEGMENT
         LH    R5,RECLL(,RECPTR)    * CALCULATE TEXT (REMAINING)
         SLR   R5,R2                  *     LENGTH
         LA    R8,24
         CR    R5,R8                  * MORE THAN 24 BYTES ?
         BNH   LESS24                 * NO, USE THIS LENGTH
*---------------------------------------------------------------------*
*      - RECORD MUST BE TRUNCATED.                                    *
*        CALCULATE TEXT LENGTH OF ALL SEGMENTS                        *
*        TO INDICATE THAT RECORD WAS TRUNCATED, CHANGE SIGN OF LENGTH *
*---------------------------------------------------------------------*
         LA    R7,0(R2,RECPTR)        * POINT TO 1ST TEXT SEGMENT
         S     R5,SUFFLEN             * SUBTRACT SEQUENCE NUMBER FROM LEN
         SLR   R3,R3                  * ZERO TEXT LENGTH COUNTER
NEXTSEG  LH    R4,0(R3,R7)            * LOAD TEXT SEGMENT LENGTH
         LTR   R4,R4                  * TEST FOR ZERO
         BZ    ALLSEGS                * SKIP IF ZERO
         AR     R3,R4                 * ACCUMULATE LENGTH
```

```
         CR    R3,R5               * COMPARE TO TOTAL LENGTH
         BL    NEXTSEG             * BRANCH IF LESS THAN TOTAL
ALLSEGS  LNR   R7,R3               * INDICATE RECORD WAS TRUNCATED
         LR    R5,R8               * ONLY USE 1ST 24 BYTES OF TEXT
LESS24   ALR   R5,R2               * GET WHOLE LENGTH TO MOVE
         LR    R0,R5               * SAVE LENGTH
         LR    R2,PBLDREC          * POINT TO TARGET AREA
         LR    R4,RECPTR           * POINT TO SOURCE AREA
         LR    R3,R5               * SET LENGTH OF SOURCE
         MVCL  R2,R4               * MOVE RECORD
         STH   R0,RECLL(,PBLDREC)  * ADJUST TARGET LENGTH
         LH    R2,RECPRELL(,RECPTR) * LOAD RECORD PREFIX LENGTH
         STH   R7,RECLL(R2,PBLDREC) * ADJUST TARGET TEXT LENGTH
         B     RECEND              * END PROCESS RECORDS 01, 03
*-------------------------------------------------------------------*
*      RECORD PROCESSOR FOR 06, 07, 31, 34, 36, 38, AND 42 RECORDS  *
*      - COPY RECORD AS IT IS                                       *
*-------------------------------------------------------------------*
RECCOPY  DS     0H                 * RECORDS TO COPY
         LR    PBLDREC,RECPTR      * POINT TO INPUT RECORD
         B     RECEND              * END PROCESS COPY-ONLY RECORDS
*-------------------------------------------------------------------*
*      RECORD PROCESSOR FOR TYPE 40 (CHECKPOINT) RECORDS            *
*      - COPY SUBTYPES 01, 03, 04, 98                               *
*      - IGNORE THE REST                                            *
*-------------------------------------------------------------------*
REC40    DS    0H                  * RECORD 40 - CHECKPOINT
         SR    R7,R7               * CLEAR WORK REGISTER
         IC    R7,RECSUBT(,RECPTR)  * GET RECORD SUBTYPE
         C     R7,TYPE03           * CNT TYPE RECORD ?
         BE    REC40USE            * YES, GO COPY IT
         C     R7,TYPE04           * SMB TYPE RECORD ?
         BE    REC40USE            * YES, GO COPY IT
         C     R7,TYPE01           * START CHECKPOINT RECORD ?
         BE    REC40USE            * YES, GO COPY IT
         C     R7,TYPE98           * END CHECKPOINT RECORD ?
         BNE   RECEND              * NO, IGNORE IT
REC40USE DS    0H                  * YES,
         LR    PBLDREC,RECPTR      *   INDICATE TO COPY RECORD
*-------------------------------------------------------------------*
*      CHECK IF ANYTHING INTERESTING FOUND                          *
*      IF SO, PUT THE RECORD                                        *
*-------------------------------------------------------------------*
RECEND   DS    0H                  * END PROCESS RECORDS
         LTR   PBLDREC,PBLDREC     * ANYTHING INTERESTING FOUND ?
         BZ    EPILOG              * NO, SKIP TO EPILOG
         LA    R1,DYNDCB           * YES, LOAD DCB ADDRESS
         USING IHADCB,R1           * LOCATE DCB
         CLC   RECLL(2,PBLDREC),DCBLRECL * IS DEFINED LRECL BIG ENOUGH?
         BH    SYNAD               * NO, TREAT AS I/O ERROR
         PUT   (1),(PBLDREC)       * YES, PUT RECORD
         DROP  R1                  * DROP BASE REG FOR DCB
         B     EPILOG              * SKIP TO EPILOG
*-------------------------------------------------------------------*
*  SYNAD EXIT - SEND A MSG, CLOSE, AND DEACTIVATE                   *
*-------------------------------------------------------------------*
SYNAD    WTO   'IMSE003 - ERROR DURING PUT TO IMSLOG',ROUTCDE=11,DESC=7
         L     ENTIND,TERMCALL     * INDICATE TO TERMINATE
         B     TERMCASE            * SKIP TO CLOSE AND TERMINATE
*-------------------------------------------------------------------*
*  END SYNAD EXIT                                                   *
*-------------------------------------------------------------------*
*-------------------------------------------------------------------*
*                                                                   *
*      TERMINATE CALL                                               *
*                                                                   *
*-------------------------------------------------------------------*
TERMCASE C     ENTIND,TERMCALL     * IS THIS THE TERMINATE CASE ?
         BNE   OTHCASE             * IF NOT SKIP ON
         OI    DYNCLOSE,X'80'      * SET HIGH ORDER BIT IN CLOS LIST
         LA    R5,DYNDCB           * LOCATE DCB
         CLOSE ((R5)),MF=(E,DYNCLOSE) * CLOSE IT
         B     EPILOG              * END OF TERMINATE CASE
*-------------------------------------------------------------------*
*                                                                   *
*      OTHER CALL, ISSUE MESSAGE AND RETURN                         *
*                                                                   *
*-------------------------------------------------------------------*
OTHCASE  DS    0H                  * START OF OTHER CASE
*                                  * ERROR, SEND A MESSAGE
         WTO   'IMSE004 - INVALID CALL TYPE',ROUTCDE=11,DESC=7
         B     NOFREEMN            * SKIP TO TERMINATE
```

```
*--------------------------------------------------------------------*
*                                                                    *
*      EPILOG                                                        *
*      - FREEMAIN STORAGE FOR TERMINATE  CALL                        *
*                                                                    *
*--------------------------------------------------------------------*
EPILOG   DS    0H
         LR    R1,R13              * POINT TO DYNAMIC AREA
         L     R13,SAVEAREA+4      * POINT TO OLD SAVE AREA
         C     ENTIND,TERMCALL     * IS THIS TERMINATION CALL ?
         BNE   NOFREEMN            * NO, DON'T FREE STORAGE
         L     R0,SIZEWORK         * PICK UP LENGTH OF DYNAMIC AREA
         FREEMAIN R,LV=(0),A=(1)   * FREE IT
         SLR   R15,R15             * GET A ZERO
         ST    R15,0(,R11)         * STORE IT INTO THE USER WORD
NOFREEMN SLR   R15,R15             * CLEAR RETURN CODE
         L     R14,12(,R13)        * RESTORE RETURN REGISTER
         LM    R0,R12,20(R13)      * RESTORE OTHER REGS
         BR    R14                 * RETURN TO CALLER
*--------------------------------------------------------------------*
*                                                                    *
*      STATIC DATA AREA                                              *
*                                                                    *
*--------------------------------------------------------------------*
         DS    0F
INITCALL DC    F'1'
NORMCALL DC    F'2'
TERMCALL DC    F'3'
SUFFLEN  DC    F'16'
TYPE01   DC    XL4'01'
TYPE03   DC    XL4'03'
TYPE04   DC    XL4'04'
TYPE42   DC    XL4'42'
TYPE98   DC    XL4'98'
         DS    0F
SIZEWORK DC    AL1(0)
         DC    AL3(((ENDWORKA-WORKAREA+7)/8)*8)
         DS    0D
         PRINT NOGENLISTDCB  DCB   MACRF=PM,DDNAME=IMSLOG,DSORG=PS,EXLST=EXITLIST, *
               SYNAD=SYNAD
LENDCB   EQU   *-LISTDCB            * LENGTH OF DCB
EXITLIST DC    XL1'85',AL3(DCBEXIT)  * DCB EXIT ADDRESS
         DCBD  DSORG=PS
*--------------------------------------------------------------------*
*                                                                    *
*      DCB EXIT                                                      *
*      - FORCE RECFM = VB                                            *
*      - ENSURE LRECL AND BLOCK SIZE ARE LARGE ENOUGH                *
*                                                                    *
*--------------------------------------------------------------------*
IMSEXIT  CSECT
DCBEXIT  DS    0H
         USING *,R15               * SET ADDRESSABILITY
         LR    DCBPTR,R1           * LOAD DCB POINTER
         USING IHADCB,DCBPTR       * LOCATE DCB
         NI    DCBRECFM,DCBRECV+DCBRECSB+DCBRECBR
*                                  * SET NOT NEEDED FLAGS OFF
         OI    DCBRECFM,DCBRECV+DCBRECBR * SET RECFM=VB
         CLC   DCBBLKSI,IMSBLOCK   * IS BLOCK SIZE
         BNL   BLOCKOK             *   GREAT ENOUGH ?
         MVC   DCBBLKSI,IMSBLOCK   * NO, SET TO USUAL SIZE
BLOCKOK  EQU   *                   * YES, OK
         CLC   DCBLRECL,TESTLREC   * IS LRECL
         BNL   LRECLOK             *   GREAT ENOUGH ?
         MVC   DCBLRECL,MAXLRECL   * NO, SET TO MAX VALUE
LRECLOK  EQU   *                   * YES, OK
         LH    R9,DCBLRECL         * LOAD LRECL
         S     R9,BDWLEN           * SUBTRACT BDW LENGTH
         CH    R9,DCBBLKSI         * LRECL &gt; BLOCK SIZE - 4 ?
         BNH   SPANNOK             * NO, SKIP ON
         OI    DCBRECFM,DCBRECSB   * YES, FORCE SPANNED RECORDS
SPANNOK  EQU   *                   * SPANNED FLAG OK
         DROP  R15                 * DROP BASE REG
         BR    R14                 * RETURN TO OPEN

MAXLRECL DC    H'32756'
IMSBLOCK DC    H'6144'
TESTLREC DC    H'6140'
BDWLEN   DC    F'4'
*--------------------------------------------------------------------*
*  END DCB EXIT                                                      *
*--------------------------------------------------------------------*
```

```
*--------------------------------------------------------------------*
*                                                                    *
*      DYNAMIC WORK AREA                                             *
*                                                                    *
*--------------------------------------------------------------------*
WORKAREA DSECT
         DS    0F
SAVEAREA DS    18F
PARMLIST DS    3FDYNDCB   DCB   MACRF=PM,DDNAME=IMSLOG,DSORG=PS,EXLST=EXITLIST
DYNOPEN  OPEN  (,),MF=L
DYNCLOSE CLOSE (,),MF=L
RECAREA  DS   0D
         DS   128CL256
ENDWORKA EQU   *
IMSEXIT  CSECT
R0       EQU   00                    EQUATES FOR REGISTERS 0-15
R1       EQU   01
R2       EQU   02
R3       EQU   03
R4       EQU   04
R5       EQU   05
R6       EQU   06
R7       EQU   07
R8       EQU   08
R9       EQU   09
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
DCBPTR   EQU   R2
RECPTR   EQU   R6
ENTIND   EQU   R10
PBLDREC  EQU   R9
*--------------------------------------------------------------------*
*                                                                    *
*      IMS RECORD MAPPING                                            *
*                                                                    *
*--------------------------------------------------------------------*
RECORD   EQU   0                  * START OF RECORD
RECLL    EQU   RECORD             * RECORD LENGTH
RECTYPE  EQU   RECORD+4           * RECORD TYPE
RECSUBT  EQU   RECORD+5           * RECORD SUBTYPE
RECPRELL EQU   RECORD+16          * TOTAL RECORD PREFIX LENGTH
         END   IMSEXIT
```

Subsection:

- “IMSLOG” on page 400

IMSLOG

The following record types are selected and written to the data set connected to the DD name IMSLOG:

Record type	Message
01	Input message.
03	Output message.
06	IMS start/stop.
07	Application accounting (MPP or BMP end).
31	Message queue get unique.
34	Message cancel.
35	Message placed on message queue.
36	Message removed from message queue.
38	Transaction reschedule.

Record type	Message
40	Checkpoint records. Only header, trailer, SMB, and CNT block records are written (subtypes 01, 03, 04, and 98 respectively).
42	IMS log header record.

To limit the size of the written log, the message text parts of the 01 and 03 records are truncated to 24 bytes. However, when this truncation occurs, the total length of all message segments is calculated and stored as a negative value in the length field of the first message segment. SLR uses this field to calculate the number of bytes transferred.

The following DCB information applies to the file IMSLOG:

Keyword	Accepted	Default
RECFM	VB	VB
BLKSIZE	6144 or greater	6144
LRECL	6140 or greater	32760 and RECFM=VBS

A program dependent on the sequence numbers of the IMS log records should not be used to process the written log data set.

LOGEDIT: Log edit user exit (DFSFLGE0 and other LOGEDIT exits)

The log edit exit routine allows you to alter the content of messages in IMS log data. This exit routine may be used to provide additional security by eliminating the logging of sensitive information under limited circumstances.

This topic contains Product-sensitive Programming Interface information.

After the message is edited, the message-related record is then logged. Even though the altered record is logged, IMS processes the original version of the message. After a subsequent restart, IMS processes the edited version of the message. If restart reschedules an edited message, the transaction might fail because of the edits.

Attention: The log edit user exit can potentially damage system information, such as the system segments in a type01 record. Use it only when no alternative exists. Test the routine rigorously before using it in a production environment.

Subsections:

- "About this routine" on page 401
- "Restrictions" on page 403
- "Communicating with IMS" on page 403

About this routine

You can write a log edit exit routine that is called before each message-related log record is written to the log. The exit overlays segments of the record data with other data. The record types that are presented to the exit are controlled through the LOGEDIT statement in the LOGGER section of the DFSDF*xxx* PROCLIB member. If you want to use the **REFRESH USEREXIT** command to add the LOGEDIT user exit, you must have selected log records to edit when IMS initialized. If no records were selected, message DFS4586E is issued and the LOGEDIT user exit is not added.

The user exit cannot directly edit log data. Instead, it returns an offset and length where data is to be changed and the address of a replacement. IMS overwrites the actual record starting at the specified offset for the specified length using the data at the address indicated for replacement data.

The user exit can specify that no alterations are to be made. Or, after specifying an edit, it can indicate that further edits are needed in the same record.

If the offset or length that is specified extends outside the data portion of the record, no editing occurs, and the user exit is notified on the next call. The user exit can assess the situation, but no further editing of the record is allowed. After the user exit returns, it is called for the next record.

This user exit is optional. No default user exit and no samples are provided. The following table shows the attributes of the log edit user exit.

Table 143. Log Edit User Exit Attributes

Attribute	Description
IMS environments	DB/DC, DBCTL.
Naming convention	You can name this exit routine DFSFLGE0 and link it into a library that is included in the STEPLIB concatenation. If DFSFLGE0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSFLGE0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.

Binding	You must bind the exit routine into IMS.SDFSRESL (or a library concatenated with it) as a separate reentrant load module. If the module is not present in the load library, the IMS logger does not load or call it.
	
	The following example demonstrates how to bind the exit routine into IMS.SDFSRESL.
	
	`//LINKIT JOB 1,MSGLEVEL=1`
	`//LINK EXEC PGM=IEWL,PARM=RENT`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL,DISP=SHR`
	`//OBJIN DD DSN=IMS.USERLIB,DISP=SHR`
	`//SYSLIN DD *`
	`  INCLUDE OBJIN(DFSFLGE0)`
	`  MODE AMODE(31),RMODE(ANY)`
	`  NAME DFSFLGE0(R)`
	`//`

Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSFLGE0. If you use multiple exit routines, specify EXITDEF=(TYPE= LOGEDIT,EXIT=(*exit_names*)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set
IMS callable services	This exit routine can use IMS callable services. This exit routine is defined to IMS as an IMS standard user exit. Exit routines that are defined to IMS receive the callable services token in the standard exit parameter list. This exit routine does not need to issue an initialization call (DFSCSII0) to use IMS callable services. This exit routine must be manually link-edited with DFSCSI00.
Sample routine location	No sample exit routine is provided.

Attributes of this routine

The log edit user exit must be written as reentrant. The user exit receives control running in 31-bit addressing mode and must return control in that mode. It is called in TASK mode, with no locks held, and in non-cross memory, non-AR mode. In an online IMS environment, the log edit user exit runs in key 7, supervisor state, in the IMS control region address space.

Calling this routine

The log edit user exit is called at each of the times described in the list that follows. The type of call is determined when IMS calls the user exit.

Initialization call
IMS calls the LOGEDIT user exit when the logger is initialized. IMS makes this call when it opens the first OLDS.

Edit record call
The log edit user exit is called immediately before the log record (OLDS or WADS) is written.

Termination call
IMS calls the LOGEDIT user exit when the logger is terminated. IMS makes this call after it closes the output log and notifies DBRC. If IMS terminates abnormally, it attempts to make this call from the log task ESTAE routine.

If IMS terminates abnormally, there might be cases when the logger cannot make the termination call to LOGEDIT. Therefore, your user exit must be able to tolerate not being called for termination.

Restrictions

The log edit user exit is subject to the following restrictions:

- This user exit should not modify any storage other than the parameter lists and associated work areas.
- All addresses, entry register contents, and parameter list contents can change from one call of the LOGEDIT user exit to the next call. One call should not depend on addresses from a prior call. The sole exception is the content of the work area indicated by SXPLAWRK, which will persist across calls.
- This user exit can call only z/OS services that it is authorized to call. It must not call any internal IMS services.

Important: The IMS logger is critical to performance. Avoid coding the exit routine to do things that could negatively affect performance in the IMS logger, such as WAITs and other z/OS services that could cause long delays before returning to your exit routine.

Communicating with IMS

IMS communicates with this user exit through the entry registers, a parameter list, and the exit registers.

Content of Registers on Entry

On entry, the user exit must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the “IMS standard user exit parameter list” on page 5
13	Address of the save area. Your user exit must not change the first three words of this save area. This save area is **not** chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this user exit.

Standard exit parameter list

This user exit uses the Version 6 standard exit parameter list. The address of the work area passed to this user exit in SXPLAWRK will be the same each time that this user exit is called.

If your LOGEDIT user exit can be called in an enhanced user exit environment, additional user exit routines might be called after your routine. When your user exit routine finds a transaction upon which to act, it can set SXPL_CALLNXTN in the byte that SXPLCNXT points to. This tells IMS to not call additional exit routines.

Function-specific parameter list on entry

The following table shows the content of the function-specific parameter list. The address of this parameter list is in the standard IMS user exit parameter list field SXPLFSPL.

Table 144. Function-specific parameter list for log edit user exit (Mapped by LGEXPL, which is included in LCDSECT)

Field	Offset	Length	Content
LGEXVERA	X'0'	X'4'	Address of parameter list version number.
LGEXTYPA	X'4'	X'4'	Address of call type field.
The remaining fields apply only to the edit record call type:			
LGEXRCDA	X'8'	X'4'	Address of log record image.
LGEXEINA	X'C'	X'4'	Address of edit instruction area.
LGEXFBKA	X'10'	X'4'	Address of feedback field.

LGEXVERA points to LGEXVER, described in the following table:

Table 145. LGEXVERA field

Field	Offset	Length	Content
LGEXVER	X'0'	X'4'	Parameter list version number.

LGEXTYPA points to LGEXCTYPE, described in the following table:

Table 146. LGEXTYPA field

Field	Offset	Length	Content
LGEXCTYP	X'0'	X'4'	Call type: • 1 = initialization call • 2 = record edit call • 3 = termination call

As shown in the preceding table, some fields apply only to the record edit call.

- LGEXRCDA points to a copy of the log record. It does not point into a log buffer.
- LGEXEINA points to the edit instructions area mapped by LGEXEI (included in LCDSECT). These fields are described in the following table.
- LGEXFBKA points to the feedback field, described in Table 148 on page 405:

Table 147. LGEXEI - edit instruction information

Field	Offset	Length	Content
LGEXFUNC	X'0'	X'4'	Functions.

Table 147. LGEXEI - edit instruction information (continued)

Field	Offset	Length	Content
LGEXFNC1	X'0'	X'1'	Functions byte 1. • X'80' = edit record using the information provided in the LGEXOFFS, LGEXLENG, and LGEXREPL fields. • X'40' = redrive exit for this record.
	X'1'	X'3'	Functions, reserved bytes.
LGEXOFFS	X'4'	X'4'	Offset for the record edit call.
LGEXLENG	X'8'	X'4'	Length for the record edit call.
LGEXREPL	X'C'	X'4'	The address of the replacement data for the record edit call

Table 148. LGEXFBKA - feedback field information

Field	Offset	Length	Content
LGEXFDBK	X'0'	X'4'	Feedback field, which includes feedback to the exit routine from IMS: • 0 = success on prior call • 4 = error in edit parameters

The user exit cannot actively edit log data. Instead, it returns an offset and length where data is to be changed and the address of a replacement. IMS overwrites the actual record starting at the specified offset for the specified length using the data at the address indicated for replacement data.

The edit instruction area is cleared before each record edit call. IMS edits the record only if LGEXEDIT is set on return to IMS. If the user exit needs to make another change in the same record, it must also set LGEXHOLD to have the same record presented again. The previous edit does not appear in the record image. IMS changes only the actual record.

If the offset or length specified extends outside the data portion of the record, no editing occurs, and the exit is called again with LGEXFDBK set to LGEXEDER. The edit instruction area will not have been cleared, so the erroneous values are present. The exit can assess the situation, but no further editing of the record is allowed. After the exit returns, it is called for the next record.

Note: The data portion of the record is defined as everything between the record type field and the clock value and sequence number at the end of the record. The logger is unaware of the significance of any part of this area. Consequently, damage to the system segments in a type01 record would go undetected and cause unpredictable results when encountered during restart. Use caution to edit only the message data.

Contents of registers on exit

Before returning to IMS, the user exit must restore all registers except for register 15, which must contain the following:

Register	Contents
15	0

Related reference

LOGGER section of the DFSDFxxx member (System Definition)

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

LOGWRT: Logger user exit (DFSFLGX0 and other LOGWRT exits)

You can write a LOGWRT user exit that is called during IMS logger execution. IMS passes the user exit all log data after the data is written to the IMS log. Your user exit can then process this data for recovery purposes.

Subsections:

About this routine

IMS calls the LOGWRT user exit with an initialization call when the logger is opened and with a termination call when the logger is closed. At these times, your exit routine can get or free any additional storage that it needs to run. IMS also calls the exit routine and passes log data to it with a write call whenever a block of data is written to the logger.

This exit routine is optional. No default and no sample are provided.

The following table shows the attributes of the LOGWRT user exit.

Attention: The IMS logger is critical to performance. Avoid coding the user exit to do something that could negatively affect performance in the IMS logger, such as WAIT and other z/OS services that could cause long delays before returning to your user exit.

Table 149. LOGWRT user exit attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL, batch.
Naming convention	You can name this exit routine DFSFLGX0 and link it into a library that is included in the STEPLIB concatenation. If DFSFLGX0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSFLGX0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.

Table 149. LOGWRT user exit attributes (continued)

Attribute	Description

Binding	You must bind the exit routine into IMS.SDFSRESL (or a library concatenated with it) as a separate **reentrant** load module. If the module is not present in the load library, the IMS logger does not load or call it.
	
	The following example demonstrates how to bind the exit routine into IMS.SDFSRESL.
	
	`//LINKIT JOB 1,MSGLEVEL=1`
	`//LINK EXEC PGM=IEWL,PARM=RENT`
	`//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD SYSOUT=A`
	`//SYSLMOD DD DSN=IMS.SDFSRESL.,DISP=SHR`
	`//OBJIN DD DSN=IMS.USERLIB.,DISP=SHR`
	`//SYSLIN DD *`
	`INCLUDE OBJIN(DFSFLGX0)`
	`MODE AMODE(31),RMODE(ANY)`
	`NAME DFSFLGX0(R)`
	`/*`

Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSFLGX0. If you use multiple exit routines, specify EXITDEF=(TYPE=LOGWRT,EXIT=(*exit_names*)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
IMS callable services	This exit routine can use IMS Callable Storage Services. This exit routine is defined to IMS as an IMS standard user exit. Exit routines that are defined to IMS receive the callable services token in the standard exit parameter list. This exit routine does not need to issue an initialization call (DFSCSII0) to use IMS callable services. You must manually bind this exit routine with DFSCSI00.
Sample routine location	No sample exit routine is provided.

Attributes of the routine

The LOGWRT user exit must be written as reentrant. The exit routine receives control running in 31-bit addressing mode and must return control in that mode. It is called in TASK mode, with no locks held, and in non-cross memory, non-AR mode. In an online IMS environment, the LOGWRT user exit runs in key 7, supervisor state, in the IMS control region address space. In batch and log recovery environments, it runs in key 8, problem state.

This information on various IMS environments is for the current release of IMS and might change in subsequent releases.

Calling the routine

The LOGWRT user exit is given control for each of the following three calls. The call type is determined by when IMS calls the routine.

Initialization call

IMS calls the LOGWRT user exit when the logger is initialized. IMS makes this call when it opens the first output log.

An initialization call (call type 1) is made for the following:

- Normal initialization (DB/DC, DBCTL, DCCTL, batch)

- Initialization of log recovery during emergency restart processing if log recovery from the write ahead data set (WADS) is required
- When the LOGWRT user exit is added using the **REFRESH USEREXIT**command
- Log Recovery utility initialization (CLS mode)
- Alternate IMS system logger initialization when the alternate IMS opens its first OLDS for output during an XRF takeover

OLDS/SLDS write call

IMS calls the LOGWRT user exit after a block of data is successfully written to the online log data set (OLDS) or the system log data set (SLDS). The OLDS is accessed in a DB/DC, DBCTL, or a DCCTL environment and in the SLDS in a batch environment.

A pointer to the data that was written is passed to the exit routine. (The blocks of data might not be presented in sequence.) All processing of the data must be completed before returning to IMS, because the data address is not valid after leaving the LOGWRT user exit.

A write call (call type 2) is made for the following:

- Normal block write (a block of data is written to the log during normal IMS processing).
- Buffer purge (the final block(s) of log data are written to the log during IMS abnormal termination).
- Log recovery during emergency restart processing (blocks of data are recovered from the WADS and written to the OLDS).
- Log Recovery utility processing when recovering entries of complete log buffers from the WADS (CLS mode).

Under some abend or error conditions, one or more blocks might be written to the log and not passed to the exit routine, or IMS might pass the same blocks to the LOGWRT user exit several times. Your exit routine must be able to tolerate both of these situations.

Termination call

IMS calls the LOGWRT user exit when the logger is terminated. IMS makes this call after it closes the output log and notifies DBRC.

A termination call (call type 3) is made for the following:

- Normal termination
- Abnormal termination (if a terminal call is possible)
- Termination of log recovery during emergency restart processing
- Log Recovery utility termination (CLS mode)

If IMS terminates abnormally, there might be cases when the logger is unable to make the termination call to the LOGWRT user exit. Therefore, your exit routine must be able to tolerate not being called for termination.

Restrictions

The LOGWRT user exit is subject to the following restrictions:

- This exit routine must not modify the log data passed to it on an OLDS/SLDS write call. It must not try to locate, access, or modify any IMS control blocks not specifically passed to it by IMS.
- All addresses, entry register contents, and parameter list contents can change from one call of the LOGWRT user exit to the next call. One call should not depend on addresses from a prior call. Similarly, this exit routine must not assume what TCB it is running under, nor that the TCB is the same from one call to the next call.
- This exit routine can call only those z/OS services that it is authorized to call. It must not call any internal IMS services.
- The Log Recovery utility does not support multiple user exit routines. The user exit must be named DFSFLGX0 for the Log Recovery utility.

Communicating with IMS

IMS communicates with this routine through the entry registers, a parameter list, and the exit registers.

Content of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the "IMS standard user exit parameter list" on page 5
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is not chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard exit parameter list

This exit routine uses the Version 6 standard exit parameter list. The address of the work area passed to this exit routine in SXPLAWRK will be the same each time that this exit routine is called.

However, the following fields are not passed to the exit when it is called from the Log Recovery utility because the data is not available:

- SXPLASCD
- SXPLRSEN
- SXPLCNXT
- SXPLFLGA

These fields are set to zero.

Function-specific parameter list on entry

The address of the function-specific parameter list is in the standard exit parameter list field SXPLFSPL. The content of the function-specific parameter list depends on whether this exit routine is called by a call type 1, 2, or 3. The following tables outline the contents of the parameter list for each of these calls.

Table 150. Function-specific parameter list for initialization call, call type 1 (mapped by LGWXPLST, which is included in LCDSECT)

Field	Offset	Length	Content
LGWXTYPE	X'0'	1	Call type: 1
LGWXENVR	X'1'	1	Environment type: **X'01'=** DB/DC online system **X'02'=** Batch IMS system (includes CICS/DLI) **X'03'=** Log Recovery utility **X'04'=** DBCTL system **X'05'=** DCCTL system

Table 150. Function-specific parameter list for initialization call, call type 1 (mapped by LGWXPLST, which is included in LCDSECT) (continued)

Field	Offset	Length	Content
LGWXFLG1	X'2'	1	Flag byte: **X'20'** **0** Not an /ERE log recovery **1** An /ERE log recovery **X'10'** **0** Not an XRF takeover **1** An XRF takeover **X'08'** **0** The LGWXTODN field does not exist **1** The LGWXTODN field does exist **X'04'** **0** The LGWXVRSN field does not exist **1** The LGWXVRSN field does exist All other flag bits are reserved.
	X'3'	1	Reserved.
LGWXTOD	X'4'	8	This field has been left here for compatibility with previous versions. The old time stamp format value is in the 00YYDDDF HHMMSSTF format.
LGWXSSID	X'C'	8	IMS subsystem ID, which is an IMSID for non-XRF online regions and DBRC=N batch regions, an RSE name for XRF online regions, and a JOB name for DBRC=Y batch regions.
LGWXBUFR	X'14'	4	Unused on this call.
LGWXBSIZ	X'18'	4	Unused on this call.
LGWXTODN	X'1C'	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time stamp format, see *IMS Version 15.2 Operations and Automation*.
LGWXVRSN	X'28'	4	This field contains the version number of the parameter list.
LGWXBF64	X'30'	8	Unused on this call.

The following table shows the parameter list for call type 2.

Table 151. Function-specific parameter list for OLDS/SLDS write call, call type 2 (mapped by LGWXPLST, which is included in LCDSECT

Field	Offset	Length	Content
LGWXTYPE	X'0'	1	Call type: 2
LGWXENVR	X'1'	1	Environment type: **X'01'=** DB/DC online system **X'02'=** Batch IMS system (includes CICS/DLI) **X'03'=** Log Recovery utility **X'04'=** DBCTL system **X'05'=** DCCTL system
LGWXFLG1	X'2'	1	Flag byte: **X'20'** **0** Not an /ERE log recovery **1** An /ERE log recovery **X'10'** **0** Not an XRF takeover **1** An XRF takeover **X'08'** **0** The LGWXTODN field does not exist **1** The LGWXTODN field does exist **X'04'** **0** The LGWXVRSN field does not exist **1** The LGWXVRSN field does exist
	X'3'	1	Reserved
LGWXTOD	X'4'	8	This field has been left here for compatibility with previous versions. The old time stamp format value is in the 00YYDDDF HHMMSSTF format.
LGWXSSID	X'C'	8	IMS subsystem ID, which is an IMSID for non-XRF online regions and DBRC=N batch regions, an RSE name for XRF online regions, and a JOB name for DBRC=Y batch regions.

Table 151. Function-specific parameter list for OLDS/SLDS write call, call type 2 (mapped by LGWXPLST, which is included in LCDSECT (continued)

Field	Offset	Length	Content
LGWXBUFR	X'14'	4	Address of IMS log block data that has been successfully written to the OLDS/SLDS. (This might be a copy of the original IMS buffer.) If the data is located above the 2 GB boundary, this field will contain X'7FFFFBAD' and a pointer to the 64-bit address of the data will be contained in field LGWXBF64.
LGWXBSIZ	X'18'	4	Length of log data, in bytes.
LGWXTODN	X'1C'	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time stamp format, see *IMS Version 15.2 Operations and Automation*.
LGWXVRSN	X'28'	4	This field contains the version number of the parameter list.
LGWXBF64	X'30'	8	This field contains the 64-bit address of the log buffer storage.

The following table shows the parameter list for call type 3.

Table 152. Function-specific parameter list for termination call, call type 3 (mapped by LGWXPLST, which is included in LCDSECT)

Field	Offset	Length	Content
LGWXTYPE	X'0'	1	Call type: 3
LGWXENVR	X'1'	1	Environment type: **X'01'=** DB/DC online system **X'02'=** Batch IMS system (includes CICS/DLI) **X'03'=** Log Recovery utility **X'04'=** DBCTL system **X'05'=** DCCTL system

Table 152. Function-specific parameter list for termination call, call type 3 (mapped by LGWXPLST, which is included in LCDSECT) (continued)

Field	Offset	Length	Content
LGWXFLG1	X'2'	1	Flag byte: **X'80'** **0** Normal termination **1** Abnormal termination **X'40'** **0** Buffer purge succeeded **1** Buffer purge failed (abend) **X'20'** **0** Not an /ERE log recovery **1** An /ERE log recovery **X'08'** **0** The LGWXTODN field does not exist **1** The LGWXTODN field does exist **X'04'** **0** The LGWXVRSN field does not exist **1** The LGWXVRSN field does exist All other flag bits are reserved.
	X'3'	1	Reserved.
LGWXTOD	X'4'	8	This field has been left here for compatibility with previous versions. The old time stamp format value is in the 00YYDDDF HHMMSSTF format.
LGWXSSID	X'C'	8	IMS subsystem ID, which is an IMSID for non-XRF online regions and DBRC=N batch regions, an RSE name for XRF online regions, and a JOB name for DBRC=Y batch regions.
LGWXBUFR	X'14'	4	Unused on this call.
LGWXBSIZ	X'18'	4	Unused on this call.
LGWXTODN	X'1C'	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time stamp format, see *IMS Version 15.2 Operations and Automation*.

Table 152. Function-specific parameter list for termination call, call type 3 (mapped by LGWXPLST, which is included in LCDSECT) (continued)

Field	Offset	Length	Content
LGWXVRSN	X'28'	4	This field contains the version number of the parameter list.
LGWXBF64	X'30'	8	Unused on this call.

For calls made during normal IMS operation, the time in the field at offset X'10' contains the start time of the current IMS system. For calls made during emergency restart log recovery, this field contains the start time of the previous IMS system, the one whose log is being recovered.

If log recovery is required during emergency restart processing, the LOGWRT user exit is called for two sets of initialization/write/termination call sequences. The first set of calls occurs during log recovery and sets a flag indicating that log recovery is processing. Only the data from the buffers (recovered from the WADS and written to the OLDS to close it) is passed. The second set of calls occurs for normal IMS processing.

Content of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain the following:

Register	Contents
15	0

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

PPUE: Partner Product exit routine (DFSPPUE0 and other PPUE exits)

The Partner Product exit routine (DFSPPUE0 or another PPUE type exit routine) is provided to allow the initialization of products that run with IMS. The exit routine can load or link one or more partner product routines.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

The Partner Product exit routine is entered immediately before IMS is ready for startup (before the DFS994I start complete message is issued). The exit routine is deleted after control returns to IMS.

Be aware that the interface to this exit routine might change in future releases of IMS.

The exit routine must reside on the library pointed to by the STEPLIB DD statement. If the exit routine exists, it is called.

The following table shows the attributes of the Partner Product exit routine.

Table 153. Partner product exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL.
Naming convention	You can name this exit routine DFSPPUE0 and link it into a library that is included in the STEPLIB concatenation. If DFSPPUE0 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSPPUE0 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.
Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSPPUE0. If you use multiple exit routines, specify EXITDEF=(TYPE= PPUE,EXIT=(*exit_names*)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
IMS callable services	This exit routine can use IMS Callable Storage Services. This exit routine is defined to IMS as an IMS standard user exit. Exit routines that are defined to IMS receive the callable services token in the standard exit parameter list. This exit routine does not need to issue an initialization call (DFSCSII0) to use IMS callable services. You must manually bind this exit routine with DFSCSI00.
Sample routine location	A sample exit named DFSPPEX0 is provided in the IMS.ADFSML data set.

Communicating with IMS

IMS uses the entry registers, a parameter list, and the exit registers to communicate with the exit routine.

Content of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “IMS standard user exit parameter list” on page 5
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is not chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard IMS user exit parameter list

This exit routine uses the Version 6 standard exit parameter list. The address of the work area passed to this exit routine in SXPLAWRK will be the same each time that this exit routine is called.

Function-specific parameter list on entry

The following table shows the content of the function-specific parameter list. The address of this parameter list is in the standard IMS user exit parameter list field SXPLFSPL.

Table 154. Function-specific parameter list for partner product exit (mapped by DFSPPUE)

Field	Offset	Length	Content
PPUEIMSD	0	4	IMS identifier
PPUEREL	4	1	IMS level
PPUETYP	5	1	IMS subsystem type
PPUEOSL	6	1	z/OS level
Reserved	7	1	

Content of registers on exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return codes	Meaning
0	Processing continues.
non-0	IMS abends with U0740. The exit routine should return this code if critical tasks do not complete successfully. If multiple DFSPPUE0 exit routines are called, the highest return code is returned to the calling program.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Restart exit routine

The Restart exit is called during all types of IMS restart.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

The Restart exit is passed a function code and a code that indicates the type of restart that is being done. The exit routines are defined to IMS using the EXITDEF parameter in the USER_EXITS section of the DFSDFxxx member; there is no default exit name. Multiple routines can be defined. The routines are called in the order that they are listed in the EXITDEF parameter.

The exit is called at the beginning of restart with a function code of x'01'. It is called after IMS has determined what type of restart is being performed and before the log is read.

This exit is called at the end of restart with a function code of X'02'. It is called immediately before the restart complete message is issued.

Attributes of this routine

Table 155. Restart exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL.
Naming convention	Any name can be used.
Including the routine	Specify EXITDEF=(TYPE=RESTART,EXITS(*exit_names*)) in the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The exit routine module or modules must be included in an authorized library that is included in the JOBLIB, STEPLIB, or LINKLIST concatenation for the system.
IMS callable services	To use callable services with this exit routine, it must be given a callable services token by IMS at the time it is given control. Check the value of the SXPLATOK field in the "IMS standard user exit parameter list" on page 5: • If SXPLATOK is zero, you cannot use callable services with this routine. • If SXPLATOK is non-zero, the callable services token is included, and you can use callable services. You can use the 256-byte work area addressed by the SXPLAWRK field to call DFSCSIF0.
Sample routine location	IMS.SDFSSMPL (member name DFSRSTX0)

Communicating with IMS

IMS uses the entry registers, a parameter list, and the exit registers to communicate with the exit routine.

Content of registers on entry

Register	Contents
1	Address of the "IMS standard user exit parameter list" on page 5
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is not chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard IMS user exit parameter list

This exit routine uses the Version 6 standard exit parameter list.

Function-specific parameter list on entry

The following table shows the content of the function-specific parameter list. The address of this parameter list is in the standard IMS user exit parameter list field SXPLFSPL.

Table 156. Function-specific parameter list for partner product exit (mapped by DFSPPUE)

Field	Offset	Length	Content
RSTX_PVER	0	4	Parameter List Version (X'00000001')

Table 156. Function-specific parameter list for partner product exit (mapped by DFSPPUE) (continued)

Field	Offset	Length	Content
RSTX_FUNC	4	4	Function Code **1** Restart Begin **2** Restart End
RSTX_TYPE	8	4	IMS restart type **1** Cold start **2** Warm start **3** Emergency restart **4** Cold comm **5** Cold base **6** Cold sys
	12	4	Reserved

Content of registers on exit

There is no requirement for exit registers and there are no defined return and reason codes.

Related reference

"Exit routine naming conventions" on page 3
Each routine name should adhere to naming conventions, including both standard z/OS conventions, and conventions that are specific to the routine.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

RECON I/O exit routine (DSPCEXT0)

The RECON I/O exit routine (DSPCEXT0) tracks changes to the RECON data set, which you can log in a journal. You can use the journal, in turn, as a trace facility, to monitor the activity of specific record types, or to write your own recovery utility for the RECON data set.

DBRC gives control to the RECON I/O exit routine (DSPCEXT0) during I/O operations to the RECON.

Subsections:

About this routine

You can code DSPCEXT0 so that it updates the journal each time a record of the data set is updated, inserted, deleted, or read. You can also record changes that are internal to the RECON access modules, such as header record extension control item changes, or the addition and deletion of multiple update control records within the data set.

You can use DSPCEXT0 when RECON access is either serial or parallel.

The following table shows the attributes of the RECON I/O exit routine.

Table 157. RECON I/O exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, and DCCTL
Naming convention	You must name this exit routine DSPCEXT0.
Binding	You must write and bind this routine as reentrant (RENT). After assembling the source code, you need to bind the object code for this module into the IMS load module DSPCINT0.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS callable services.
Sample routine location	The IMS.ADFSSRC data set contains member name DSPCEXT1, which you can modify to provide support for both BPE and non-BPE based DBRC environments. DSPCEXT1 must be linked as DSPCEXT0.

You must write and bind DSPCEXT0 as reentrant. It is entered from DBRC in 31-bit addressing mode and must return to DBRC in 31-bit addressing mode. All parameters and data areas supplied to DSPCEXT0 by DBRC are located above the 16 MB line. In addition, load module DSPCINT0, in which DSPCEXT0 is located, resides above the line. Note that due to the residency of DSPCINT0, unless you specify otherwise, GETMAIN will acquire storage above the 16 MB line when issued for DSPCEXT0.

No further calls to DSPCEXT0 occur if it terminates abnormally. DBRC recovers the termination and carries on normally thereafter.

Calling the routine

Control is passed to the RECON I/O exit routine whenever a RECON record has been successfully read, written, or modified on COPY 1 of the RECON data set, not necessarily for every physical I/O operation. Changes to the header record extension also cause the RECON I/O exit routine to be called.

When RECON access is parallel, the RECON data set can be accessed by multiple DBRC instances concurrently. In this case, multiple instances of the RECON I/O exit routine can be invoked concurrently.

With serial access, the user can rely on all updates written to the RECON data set. If an error occurs, and the update is backed out by DBRC, the exit is called for all the updates made during backout. If the exit is used to mirror updates, the exit can immediately make the equivalent updates to a mirror data set.

With parallel access, the backout of data is not done by DBRC, which means that the exit is not called for the backout updates. Updates made during a given series should not be considered hardened in the RECON data set until a commit call is made. If the exit is used to mirror updates, it must either be capable of backing out the updates it mirrors, or it must collect all updates for a given series and only mirror them if the exit is called with a commit call.

Whenever a record of data is inserted, updated, deleted, or read, this routine is called after the call or change is made to the RECON data set. For each insert, delete, and read call, the routine receives a copy of the inserted, deleted, or read record, respectively. For each update call, the routine receives a copy of the record as it appeared both before and after it was updated. For delete and update calls, the copy of

the record read must be incomplete if DBRC is unable to locate all segments for that record. In this case, byte 2 of word 17 in the I/O exit parameter list is set to X'40'.

The records passed to the exit routine are in the format of the release level of the RECON data set, and rather than the release level of the DBRC that calls the exit. In order for the DBRCs of multiple IMS systems at different release levels to coexist, the RECON data set must be at the level of the highest level system. An indication of the RECON data set release level exists in the parameter list that is passed to the exit. When the RECON is upgraded to a new release, the exit routine can use both the old release format and the new release format. During the upgrade process, the release level in the parameter list shows the old release level. A flag in the parameter list indicates that an upgrade is in progress.

The release level of the RECON can change from one Begin Series call to another. Except during the upgrade process, the release level does not change between the Begin Series call and the Terminate Series call.

Any modifications to storage that this routine makes must be made to storage that is obtained by the routine, not to the data areas pointed to by DBRC or IMS or to those contained within the routine itself.

Each series of I/O accesses that DBRC makes to the RECON data set is indicated to the routine by a Begin Series call. When the series of I/O operations is complete, the routine receives a Terminate Series call.

Performance recommendations

While this routine is running, the RECON data set is reserved so that no other jobs can access RECON records. To minimize the affect that this routine's execution has on your system's performance, you need to:

- Limit the I/O operations that the routine itself performs and simplify the routine's functions to make efficient use of processing time.
- Be sure that any resources needed solely by the routine (that is, those not needed by DBRC/IMS in general) are immediately available to z/OS when DBRC is initialized and in control. You should therefore avoid operations that can put the routine, and therefore DBRC, in a prolonged wait state (for example, the ENQUEUE/DEQUEUE of resources that cannot be readily accessed by the routine or write to operator messages that require waiting for a reply).
- Be aware that with parallel RECON access, the RECON data set is not reserved. In addition, multiple instances of the RECON I/O exit routine can be invoked concurrently.

DBRC enables the size of a record in the RECON data set to not be limited by the defined RecordSize. DBRC divides its own records into segments, each of which fits into a single Control Interval (CI) and is sent by VSAM as a complete record. Segmenting allows a logical RECON record to be as large as 16 777 215 bytes. The RECON I/O exit routine will be presented with complete, unsegmented logical records.

To minimize the performance impact that the routine's execution has on DBRC, the routine spools its copy of RECON data records to a data set (specified by a DD statement with the name DBRCDATA) for later offline processing outside the DBRC environment. Any data sets that your routine references need to be accessed by DD statements as well.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of a standard z/OS parameter list. This list consists of a fullword with the high-order bit ON, indicating the last entry in the list. The remaining bits comprise the address of a data area containing the RECON I/O parameter list (DSPRIOX).

Register	Contents
13	Address of save area. The exit routine must not change the first three words.
14	Return address to the calling RECON access routine.
15	Entry point of exit routine.

Description of parameters

This routine receives the parameter list from the calling RECON access module at the first Begin Series call for a job. The parameter list points to the same data area for all subsequent calls for that job.

The data area pointed to by the parameter list is 24 words (96 bytes) long and starts on a fullword boundary. Words 9 through 16 of the list are free to be used by the exit routine and remain unchanged by DBRC after the first Begin Series call. They initially contain all zeros.

The first byte of word 17 of the list indicates the release level of the RECON in hexadecimal format. The following table lists RECON release levels by IMS version:

Version	RECON release level in hexadecimal format
IMS Version 13	X'D1'
IMS Version 12	X'C1'
IMS Version 11	X'B1'

Byte 2 of word 17 contains flags. Bytes 3 and 4 of Word 17, and Words 22 through 24 are reserved for future use.

The following tables list the exit parameter list at various exit points in the routine.

Table 158. Begin Series parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 1 - "Begin Series" A binary **1**, indicating a Begin Series call to this routine as a result of a RESERVE function having been performed on the RECON data set.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. • All calls to the exit for this series, including the Terminate Series call, will have this request token. • For serial RECON access, the token is the RESERVE sequence number from the control record extension. This number is incremented by one in the control record extension each time DBRC completes a successful RESERVE of the RECON data set. • For parallel RECON access, the token is a store-clock (STCK) value captured before the RECON I/O exit is invoked.

Table 158. Begin Series parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_CHANGECNT	X'10'	X'04'	Input	Changed record count. • For serial RECON access, this is the changed record count from the control record extension. The changed record count is a 32-bit logical value that can eventually wrap back to zero. This is the count as of the last DEQUEUE function that DBRC performs, or that value plus one if the last DBRC abended. A change to the RECON data set has occurred if an ENQUEUE sequence detects that the last DBRC abended. For more information about the changed record count, see the "Terminate Series" exit call in this topic. • For parallel RECON access, the count is always zero. The RECON I/O exit routine interprets a zero count to mean that parallel access is in effect.
	X'14'	X'2C'	None	Reserved.
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. For IMS Version 12, the value is X'C1'. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	• For serial RECON access, binary zeroes. • For parallel RECON access, the DBRC instance token. This token is a binary value that can be used to distinguish the calls in a given series in case two DBRC instances present the same STCK value (request token). The instance token is unique across currently executing DBRC instances.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 159. Insert record parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 3 - "Insert". A binary 3, indicating an insert call to this routine as a result of a record having been inserted into the RECON data set.
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
RIOX_RECORDLEN	X'10'	X'04'	Input	Length of the record that has been inserted.
RIOX_RECORDADR	X'14'	X'04'	Input	Address of the record that has been inserted.
	X'18'	X'28'	None	Reserved.

Table 159. Insert record parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 160. Update record parameter list

Field Name	Offset	Length	Field Usage	Meaning or content
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	A binary **4**, indicating an update call to this routine as a result of a record having been updated on the RECON data set.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. Unchanged from the Begin Series call.
RIOX_OLDRECLEN	X'10'	X'04'	Input	Length of the record image before update.
RIOX_OLDRECADR	X'14'	X'04'	Input	Address of a copy of the record as it appeared before the update.
RIOX_NEWRECLEN	X'18'	X'04'	Input	Length of the replacement record.
RIOX_NEWRECADR	X'1C'	X'04'	Input	Address of the replacement record.
	X'20'	X'20'	None	Reserved.
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is ON if the record, before being changed, had a missing segment. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 161. Delete record parameter list

Field Name	Offset	Length	Field Usage	Meaning or content
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 5 - "Delete". A binary 5, indicating a delete call to this routine as a result of a record having been deleted from the RECON data set.

Table 161. Delete record parameter list (continued)

Field Name	Offset	Length	Field Usage	Meaning or content
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
RIOX_OLDRECLEN	X'10'	X'04'	Input	Length of the record that has been deleted.
RIOX_OLDRECADR	X'14'	X'04'	Input	Address of the record that has been deleted.
	X'18'	X'28'	None	Reserved.
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 162. Read record parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 6 - "Read". A binary 6, indicating a read call to this routine as a result of a record having been read from the RECON data set.
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
RIOX_OLDRECLEN	X'10'	X'04'	Input	Length of the record that has been read.
RIOX_OLDRECADR	X'14'	X'04'	Input	Address of the record that's been read.
	X'18'	X'28'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 163. Commit request parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"

Table 163. Commit request parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_FUNC	X'04'	X'04'	Input	Function Code 7 - "Commit". A binary 7, indicating a commit call to this routine. The call results from previous updates (including inserts and deletes) for this current series being committed to the RECON data set. This call is made only for parallel RECON access.
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
	X'10'	X'30'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains flags that are defined as follows: • Bit 0 is ON when upgrade is in progress • Bit 1 is OFF • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 164. Backout request parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	A binary **8**, indicating a backout call to this routine. The call results from previous updates (including inserts and deletes) for this current series being backed out of the RECON data set. This call is made only for parallel RECON access.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. Unchanged from the Begin Series call.
	X'10'	X'30'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 165. Terminate Series parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"

Table 165. Terminate Series parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_FUNC	X'04'	X'04'	Input	Function Code 2 - "Terminate Series" A binary **2**, indicating a Terminate Series call to this routine. This call occurs at the end of processing a DBRC request. For serial RECON access, the DEQUEUE for the RECON has been performed.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. Unchanged from the Begin Series call.
RIOX_CHANGECNT	X'10'	X'04'	Input	Final changed record count. • For serial RECON access, this is the final changed record count as it now appears on the control record extension. The changed record count is a 32-bit logical value that can eventually wrap back to zero. Either the count is the same as the Begin Series call value, or it is that value plus one if any change has been made (other than to the record extension itself) to the RECON data set since the Begin Series call. By monitoring the value of this counter between its value here and the next Begin Series exit call, you can detect changes made to the RECON data set by other occurrences of DBRC. • For parallel RECON access, the count is always zero. With parallel access, you cannot detect when changes have been made to the RECON by other DBRC instances.
	X'14'	X'2C'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	• For serial RECON access, binary zeroes. • For parallel RECON access, the DBRC instance token. This token is a binary value that can be used to distinguish the calls in a given series in case two DBRC instances present the same STCK value (request token). The instance token is unique across currently executing DBRC instances.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Contents of registers on exit

Before returning to DBRC, the exit routine must restore all registers except register 15, which must contain one of the following return codes.

The following table reflects the register contents for non-BPE based DBRC exit routines.

Return code	Meaning
0	The exit routine is called.

Return code	Meaning
nonzero	No further calls to this exit routine are made.

Related concepts

Initializing and maintaining the RECON data sets (System Administration)

Related reference

"BPE-based DBRC user exit routines" on page 529
The BPE-based DBRC user exit routines enable you to run the existing DBRC user exit routines in a BPE (Base Primitive Environment).

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"RECON I/O exit routine " on page 534
A BPE-based DBRC gives control to the RECON I/O exit routine during I/O operations to the RECON data set. This exit routine performs the function of DSPCEXT0 for BPE-based DBRCs.

Minimizing impact to system performance

You can minimize the impact of running the RECON I/O exit routine (DSPCEXT0) on system performance by following three recommended steps.

While this routine is running, the RECON data set is reserved so that no other jobs can access RECON records. To minimize the affect that this routine's execution has on system performance:

1. Limit the I/O operations that the routine completes and simplify the functions of the routine to use processing time efficiently.
2. Ensure that any resources that are needed solely by the routine (needed by DBRC or IMS) are immediately available to z/OS when DBRC is initialized and in control. Avoid operations that can put the routine, and therefore DBRC, in a prolonged wait state (for example, the enqueue or dequeue of resources that cannot be readily accessed by the routine, or write to operator messages that require waiting for a reply).
3. Be aware that with parallel RECON access, the RECON data set is not reserved. In addition, multiple instances of the DSPCEXT0 routine can be invoked concurrently.

DBRC divides its own records into segments, each of which fits into a single Control Interval (CI) and is sent by VSAM as a complete record. Segmenting allows a logical RECON record to be as large as 16 777 215 bytes. The DSPCEXT0 exit routine will be presented with complete, unsegmented logical records.

RASE: Resource Access Security user exit (DFSRAS00 and other RASE exits)

The Resource Access Security user exit (RASE) authorizes IMS resources such as transactions, PSBs, or output LTERM names. This user exit is called after the SAF interface is called.

Subsections:

About this routine

This user exit is called during IMS dependent region initialization, or during CCTL, ODBA, or ODBM connection, to allow the user to instruct IMS to perform one of the functions described in the return codes section. For example, this user exit can terminate a connection with a user abend code 437.

This user exit is called to perform pre-authorization processing and can instruct IMS to skip PSB or transaction authorization processing for any of the following thread instances:

- IMS dependent regions, CCTL connections

  The pre-authorization process is performed only if the exit returns with return code 4 or 24 from initialization or connection processing, and ISIS=R or ISIS=A is specified.
- ODBA connections

  The pre-authorization process is performed only if the exit returns with return code 4 or 24 from connection processing, and one of the following conditions is true:

  – ODBASE=Y is specified.
  – ODBASE=N and either ISIS=R or ISIS=A is specified.
- ODBM connections

  The pre-authorization process is performed only if the exit returns with return code 4 or 24 from connection processing, and one of the following conditions is true:

  – ODBMSECURE=R or ODBMSECURE=A is specified.
  – ODBMSECURE=I or ODBMSECURE is not specified. ODBM runs without RRS (RRS=N). Either ISIS=R or ISIS=A is specified.
  – ODBMSECURE=I or ODBMSECURE is not specified. ODBM runs with RRS (RRS=Y). ODBASE=Y is specified.
  – ODBMSECURE=I or ODBMSECURE is not specified. ODBM runs with RRS (RRS=Y). ODBASE=N is specified. Either ISIS=R or ISIS=A is specified.

If ISIS=A, ISIS=C, ODBMSECURE=A, or ODBMSECURE=E is specified, the RASE user exit is required at IMS initialization. If the exit is not available during IMS initialization, IMS terminates with a user abend code 107, subcode x'04'. The RASE user exit is optional if none of ISIS=A, ISIS=C, ODBMSECURE=A, and ODBMSECURE=E is specified.

The RASE user exit can be added or deleted using the **REFRESH USEREXIT** command. If you delete the RASE user exit with the **REFRESH USEREXIT** command, DFS4585W message is issued. The ISIS and ODBASE values are included in the message text.

Specify the requirement to call the SAF interface and user exit for ODBM threads using the ODBMSECURE parameter at system initialization.

This user exit does not support callable services.

The following table shows the attributes for the Resource Access Security user exit.

Table 166. Resource Access Security user exit attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	You can name this exit routine DFSRAS00 and link it into a library that is included in the STEPLIB concatenation. If DFSRAS00 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSRAS00 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter.
Binding	You must write the exit routine as reentrant.

Table 166. Resource Access Security user exit attributes (continued)

Attribute	Description
Including the routine	The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSRAS00. If you use multiple exit routines, specify EXITDEF=(TYPE=RASE,EXIT=(exit_names)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
IMS callable services	This routine is not eligible for IMS callable services.
Sample routine location	IMS.ADFSSMPL

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the user exit.

Contents of registers on entry

On entry, the user exit must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “IMS standard user exit parameter list” on page 5
13	Address of the save area.
14	Return address of IMS.
15	Entry point address of user exit.

IMS standard user exit parameter list (SXPL)

This user exit uses the Version 6 standard exit parameter list. The address of the work area passed to this user exit in SXPLAWRK can be different each time that this user exit is called.

If your RASE user exit can be called in an enhanced user exit environment, additional user exit routines can be called after your routine. When your user exit routine finds a transaction upon which to act, it can set SXPL_CALLNXTN in the byte that SXPLCNXT points to. This tells IMS to not call additional exit routines.

Resource Access Security exit routine parameter list

The following table shows the function-specific parameter list that is mapped by DFSRASL.

Table 167. Function-specific parameter list mapped by DFSRASL

Field	Offset	Length	Content
RASLVER	0	4	Version number for DFSRASL

Table 167. Function-specific parameter list mapped by DFSRASL (continued)

Field	Offset	Length	Content
RASLFUNC	4	1	Reason for entering the RASE user exit: **X'01'** Initialization **X'02'** Authorize transaction (MPP, JMP) **X'03'** Authorize PSB (IFP, non-message-driven BMP, JBP, DRA/CCTL|ODBA) **X'04'** Authorize transaction and PSB (message-driven BMP) **X'05'** Authorize PSB and output LTERM (non-message-driven BMP, JBP) **X'06'** Authorize PSB and output transaction (non-message-driven BMP, JBP) **X'07'** Dependent region initialization **X'08'** AER/ODBA thread initialization or connection **X'09'** CCTL/DBCTL thread initialization or connection **X'0A'** Pre-authorize PSB or transaction for dependent region or CCTL/AER thread. This function skips normal PSB or transaction authorization for functions X'02' to X'06' that would normally be invoked after the pre-authorization processing. **X'0B'** ODBM connection initialization **X'0C'** ODBM thread APSB PSB authorization

Table 167. Function-specific parameter list mapped by DFSRASL (continued)

Field	Offset	Length	Content
RASLENVR	5	1	Type of dependent region for which exit was called: **X'01'** MPP **X'02'** IFP **X'03'** Message-driven BMP **X'04'** Non-message-driven BMP **X'05'** JMP **X'06'** JBP **X'07'** DRA thread from a CCTL task **X'08'** DRA thread from an ODBA task **X'09'** CPI-C MPP **X'0A'** ODBM thread
RASFLG1	6	1	Flag byte: **X'01'** ODBASE=Y specified **X'02'** ISIS=C specified **X'04'** ISIS=R specified **X'08'** ODBMSECURE=R **X'10'** ODBMSECURE=E **Note:** If bit X'04' and bit X'02' are both on, ISIS=A is specified for the IMS system. **Note:** If bit X'08' and bit X'10' are both on, ODBMSECURE=A is specified for the IMS system.
RASLESV	7	1	Reserved
RASLTRAN	8	8	Transaction code (for BMPs, from IN= if message driven, and from OUT= if non-message-driven)
RASLTSRC	16	4	SAF return code for transaction
RASLTRRC	20	4	RACF (or equivalent) return code for transaction
RASLTRRS	24	4	RACF (or equivalent) reason code for transaction
RASLPSB	28	8	PSB name

Table 167. Function-specific parameter list mapped by DFSRASL (continued)

Field	Offset	Length	Content
RASLPSRC	36	4	SAF return code for PSB
RASLPRRC	40	4	RACF (or equivalent) return code for PSB
RASLPRRS	44	4	RACF (or equivalent) reason code for PSB
RASLLTRM	48	8	Output LTERM name
RASLLSRC	56	4	SAF return code for LTERM
RASLLRRC	60	4	RACF (or equivalent) return code for LTERM
RASLLRRS	64	4	RACF (or equivalent) reason code for LTERM
RASLECB	68	4	ECB address
RASLTCDE	72	8	Input transaction code
RASLPGM	80	8	Program name
RASLUSID	88	8	User ID of dependent region
RASLGRPN	96	8	Group name
RASLSSTY	104	1	IMS environment flag: **X'01'** DB/DC system **X'02'** DCCTL system **X'03'** DBCTL system
RASLROLE	105	1	XRF role flag: **X'01'** XRF active IMS **X'02'** XRF alternate IMS
RASLMVSL	106	1	z/OS version and release on which IMS was generated
RASLUIDI	107	1	User ID indicator: **RASLUIDU** User ID in RASLUSID field **RASLUIDL** LTERM in RASLUSID field **RASLUIDP** PSB name in RASLUSID field **RASLUIDO** Other in RASLUSID field
RASLIMSI	108	8	IMS subsystem identifier
RASLIMSL	116	4	IMS version and release
RASLJOBN	144	8	Job name for the dependent region or CCTL/AER address space
RASLSSNM	152	8	Subsystem name for the CCTL/AER thread

Notes:

1. When the RASE user exit is used to authorize two resources, the exit routine is called twice: once for each resource. On the first call, one resource field (RASLTRAN, RASLPSB, or RASLLTRM) contains the resource name and the other resource field contains binary zeros. If the first call is successful, on the second call, the resource field that contained zeros in the first call contains the resource name and the other resource field that contained the resource name contains binary zeros.

   For example, to authorize a PSB and output LTERM, the first call is made with the RASLPSB containing the PSB name and RASLLTRM containing binary zeros. On the second call, RASLPSB contains zeros and RASLLTRM contains the LTERM name.

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains one of the following return codes:

Return code	Meaning
0	Resources valid for this user
4	IMS must perform pre-authorization processing for PSB or transaction authorization. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'07', X'08', or X'09'.
8	Resources invalid for this user. For function codes X'07', X'08', and X'09' in the RASLFUNC field, this return code instructs IMS to issue a DFS2854A message and terminate the dependent region or CCTL/AER thread with ABENDU0437.
12	IMS must skip the subsequent PSB or transaction authorization processing for this instance of this thread. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'0A'.
16	IMS must skip all subsequent PSB or transaction authorization processing for all instances of this thread. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'07', X'08', or X'09'.
20	IMS must skip the subsequent user authorization processing of the IMS APPL ID during dependent region initialization or CCTL/AER thread connection. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'07', X'08', or X'09'. If this return code is specified, IMS will skip the SAF FASTAUTH call that is normally performed for PSB or transaction authorization when ISIS=A or R is specified for the IMS system.
24	IMS must perform authorization processing as indicated in both return code 4 and return code 20. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'07', X'08', or X'09'.
28	IMS must perform authorization processing as indicated in both return code 16 and return code 20. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'07', X'08', or X'09'.
32	IMS must perform the subsequent PSB authorization processing for this instance of this thread, but must skip the subsequent transaction or LTERM authorization processing that is normally performed. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'0A' for a message-driven BMP or for a BMP/JBP with the OUT= parameter specified.

Return code	Meaning
36	IMS must perform the subsequent transaction or LTERM authorization processing for this instance of this thread, but must skip the subsequent PSB authorization processing that is normally performed. IMS honors this return code instruction only when the function code in the RASLFUNC field is X'0A' for a message-driven BMP or for a BMP/JBP with the OUT= parameter specified.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS standard user exit parameter list" on page 5
Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

System Definition Preprocessor exit routine (input phase) (DFSPRE60)

System Definition Preprocessor exit routine (Input Phase) can be used to alter, insert, or delete data from stage 1 input before the Preprocessor record scan occurs.

Subsections:

About this routine

DFSPRE60 is a System Definition Preprocessor exit routine. This routine is entered following the reading each stage 1 input record.

Control passes to DFSPRE60 after each stage 1 input record is read from SYSIN, and also after each record (if any) is read from SYSLIB, but before any such records are scanned by the preprocessor. Stage 1 data is presented to DFSPRE60 exactly as read by the preprocessor. Data can be altered, inserted, or deleted during this exit routine phase. However, the alterations, insertions, or deletions are not passed to Stage 1.

Related Reading: For a description of the system definition preprocessor, see *IMS Version 15.2 System Definition*.

You can also use this exit routine for construction of tables or for verification as required by installation practices. For example, if you want to check if transaction codes (which are also IMS command keywords) are accidentally defined, this routine can insert TRANSACT macros for each IMS command keyword. The results of changes appear in the listing produced by the preprocessor, unless the exit routine returns a return code of 4. However, no update of the original input is performed.

If this exit routine is used, you must specify Y as the first positional parameter in the parameter field of the EXEC card. The routine module must be named DFSPRE60, and it must reside on the library pointed to by the STEPLIB DD statement. If concatenation is used, the libraries are searched according to z/OS rules.

The following table shows the attributes of the System Definition Preprocessor (Input Phase) exit routine.

Table 168. System definition preprocessor exit routine (input phase) attributes

Attribute	Description
IMS environments	DB/DC, DCCTL, and DBCTL (with modifications).
Naming convention	You must name this exit routine DFSPRE60.

Table 168. System definition preprocessor exit routine (input phase) attributes (continued)

Attribute	Description
Binding	You **must** bind this routine with RMODE=24; otherwise, an abend can occur.
Including the routine	No special steps are required to include this routine.
IMS callable services	IMS callable services are not applicable for use with this exit routine.
Sample routine location	IMS.ADFSSMPL (member name DFSPRE60).

Communicating with IMS

IMS communicates with the System Definition Preprocessor exit routine through the entry and exit registers.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
10	Address of vector table.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Description of parameters

The parameters are listed in parameter list format and vector table format.

Parameter List Format

```
A(INPUT) (or A(0) at end of file)
  INPUT: These stage 1 source statements reside on the data set
         pointed to by the DD statement SYSIN or the copy members
         from SYSLIB.
```

Vector Table Format

```
A(DFSPRE30)  -  Entry point to QUICKSORT routine
A(DFSPRE40)  -  Entry point to duplicate check routine
A(DFSPRE50)  -  Entry point to cross check routine
A(0)         -  Reserved for user
```

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return code	Meaning
X'00'	Normal return; record is processed.
X'04'	Do not process record. Record is not printed.
X'08'	Return to exit routine. Pass control to this exit routine before reading another input record. The input buffer, as modified by the exit routine, is processed as an input record.
X'0C'	Do not call this exit routine again. Record is processed.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample system definition preprocessor exit routine

Use the sample system definition preprocessor exit routine to alter, insert, or delete data from stage 1 input before the Preprocessor record scan occurs.

The input statements are scanned for a comment card indicating that TRANSACT macros are to be read from a user file, and passed to the preprocessor in the input buffer area whose address is passed on entry. While the user file is read, the exit routine returns a code of X'08', indicating that the preprocessor is to continue calling the exit routine, instead of reading input records from the SYSIN file.

When the end of file is reached on the user file, the exit routine returns a code of X'0C', indicating that the exit routine is not to be invoked again. The statements passed to the preprocessor are handled identically to the statements on the SYSIN file. However, these statements are not written out to the SYSIN file for later processing by Stage 1.

System Definition Preprocessor exit routine (name check complete) (DFSPRE70)

The System Definition Preprocessor exit routine (name check complete) DFSPRE70, can be used to build tables that contain resource names that have been cross-checked.

Subsections:

About this routine

This exit routine is entered when all cross-checking of resource names is completed.

To track the changes between IMS system definitions, you can access all tables constructed by the preprocessor, and write them on files for input to a user program.

If this exit routine is used, you must specify Y as the second positional parameter in the parameter field on the EXEC card. The exit module, which must be named DFSPRE70, must reside on the library pointed to by the STEPLIB DD statement. If concatenated, the libraries are searched according to z/OS rules. The processing in this exit routine does not affect the previous preprocessor results or error messages.

The following table shows the attributes of the System Definition Preprocessor exit routine (name check complete).

Table 169. System definition preprocessor exit routine (name check complete) attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSPRE70.
Including the routine	No special steps are required to include this routine.
IMS callable services	IMS callable services are not applicable for use with this exit routine.
Sample routine location	IMS.SDFSSMPL (member name DFSPRE70). In the sample routine, the source name tables are written out for later processing by user programs. The end of each exit routine is indicated by the insertion of high values (X'FF').

Communicating with IMS

IMS communicates with the System Definition Preprocessor exit routine through the entry and exit registers.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
10	Address of vector table.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Description of parameters

These are the addresses and sizes of tables of resource names. The contents of the count field (fullword) determine the usefulness of the table field. If the count field is zero, the contents of the table field are not guaranteed and should be ignored. If the count field is nonzero, the table field contains the address of the table of resource names.

Table 170. Parameter list

Resource address	Contents
A(Table1)	DBD names
A(Count1)	Number of entries in Table1
A(Table2)	PSB names
A(Count2)	Number of entries in Table2
A(Table3)	Transaction codes
A(Count3)	Number of entries in Table3
A(Table4)	MSNAME linknames
A(Count4)	Number of entries in Table4
A(Table5)	VTAM node names
A(Count5)	Number of entries in Table5
A(Table6)	LTERM names
A(Count6)	Number of entries in Table6
A(Table7)	Subpool names
A(Count7)	Number of entries in Table7
A(Table8)	Routing codes
A(Count8)	Number of entries in Table8
A(Table9	Physical link names
A(Count9)	Number of entries in Table9

Table 170. Parameter list (continued)

Resource address	Contents
A(Table10)	Remote system VTAM node names
A(Count10)	Number of entries in Table10
A(Table11)	MSLINK partner IDs
A(Count11)	Number of entries in Table11

Table 171. Vector table format

Resource address	Contents
A(DFSPRE30)	Entry point to QUICKSORT routine
A(DFSPRE40)	Entry point to duplicate check routine
A(DFSPRE50)	Entry point to cross check routine
A(0)	Reserved for user

Contents of registers on exit

Before returning to IMS, the exit routine must restore all registers. Register 15 can contain a return code, but the preprocessor ignores it.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Type-1 Automated Operator exit routine (DFSAOUE0)

The Type-1 Automated Operator exit routine (DFSAOUE0) is called continuously for operator-entered commands from terminals, APPC and OTMA devices, and the command responses to operator-entered commands. The AO exit routine intercepts these messages before IMS sends the system message, runs the command, or sends the command response.

This topic contains Product-sensitive Programming Interface information.

You can write two types of Automated Operator (AO) exit routines. The AO exit routine that is described in this topic is called a *type-1 AO exit routine*. It can be used in the DB/DC and DCCTL environments.

The other AO exit routine (DFSAOE00 or another AOIE type exit routine) is called a *type-2 AO exit routine* and can be used in the DB/DC, DCCTL, and DBCTL environments.

If both DFSAOUE0 and the AOIE type exit routine are provided in a DB/DC or DCCTL environment, the AOIE type exit routine is called first. The AOIE type exit routine determines which exit routine handles the message, command, or command response.

Subsections:

About this routine

System messages that are destined for terminals other than the master terminal operator (MTO) and certain commands and command responses do not cause IMS to call this exit routine.

You can write the exit routine to handle both single and multisegment messages, and to perform the following functions:

- Ignore selected segments or an entire message.
- Send a copy of a system message, command, or command response to an alternate destination.
- Send a new message to an alternate destination for a system message, command, or command response.
- Change a system message.
- Change a system message and send a copy to an alternate destination.
- Change a copy of a command or command response and send the copy to an alternate destination.
- Delete a system message.
- Delete a system message and send a copy to an alternate destination.
- Request the edited command buffer (when the input is a command).

The following table shows the attributes for DFSAOUE0.

Table 172. Automated operator exit routine attributes (DFSAOUE0)

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSAOUE0.
Binding	DFSAOUE0 is a stand-alone 31-bit module that you must provide. **Recommendation:** Code the exit routine module as reentrant. You must manually link-edit the routine with DFSCSI00 to include the routine.
Including the routine	You specify DFSAOUE0 by linking it in the IMS.SDFSRESL concatenation as a stand-alone module. DFSAOUE0 is then loaded and called. If you specify both DFSAOUE0 and the AOIE type exit routine (the other AO exit routine), both are loaded. The AOIE type exit routine is called first and can either process the message, command, or command response, or it can return a code indicating DFSAOUE0 should be called to do the processing instead.
IMS callable services	DFSAOUE0 can use callable services for storage and control block functions. To use callable services, issue an initialization call (DFSCSII0) to get the callable services token and a parameter list in which to build the function-specific parameter list for the callable service you want to use. Use the ECB found in register 9 for the DFSCSII0 call.
Sample routine location	IMS.ADFSSMPL The AO exit routine can work with an AO application. The exit routine can insert a message to an alternative destination that is an AOI transaction without using an AO application. (For more information, see "Using the Sample AO Application (UETRANS) " in *IMS Version 15.2 Operations and Automation*.) The AO exit routine and the AO application used together serve as an example of how to use AOI.

How this routine processes messages

The following figure shows how an AO exit routine intercepts a command that is entered from a master terminal.

1. The command is entered.

2. The command controller passes a copy of the command to the exit routine before it runs the command. The exit routine can send a copy of the command to any destination (LTERM or transaction).
3. The exit routine returns to the command controller, where the command is run.
4. When the response to the command is returned to the command controller, a message is generated for the master terminal.
5. Before the message is sent, the exit routine receives control and can route a copy of the message to any destination (LTERM or transaction).
6. The message is then sent to the master terminal.

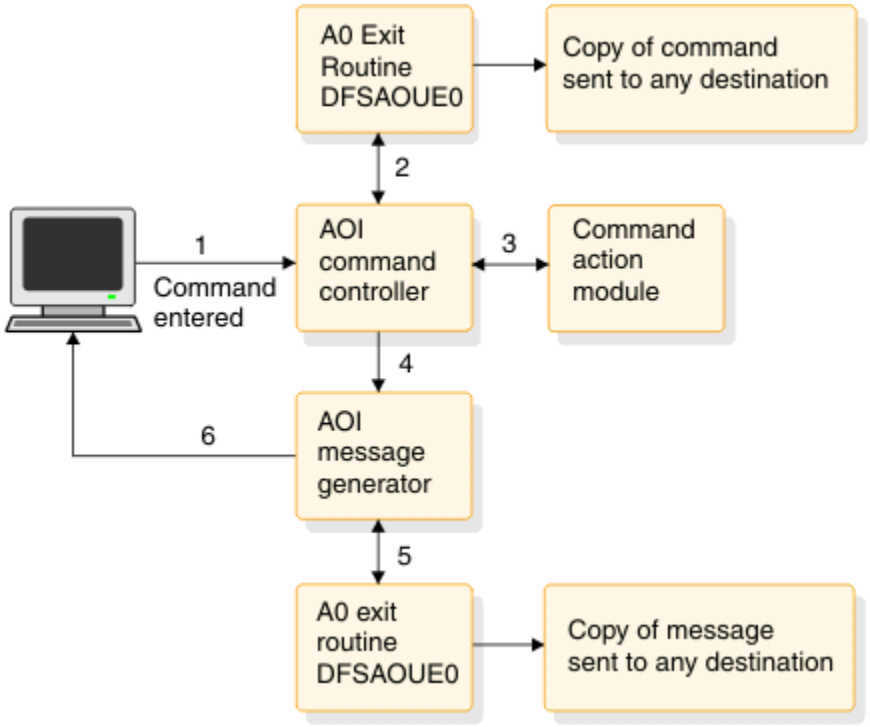


Figure 23. Processing when a command is entered at the terminal

The following figure shows how an exit routine processes a system message that is destined for the master terminal. When a system message is generated, the exit routine receives a copy of the message before the message is sent to the master terminal. The exit routine can route a copy of the message to any destination. It can alter or delete any segment of the message.

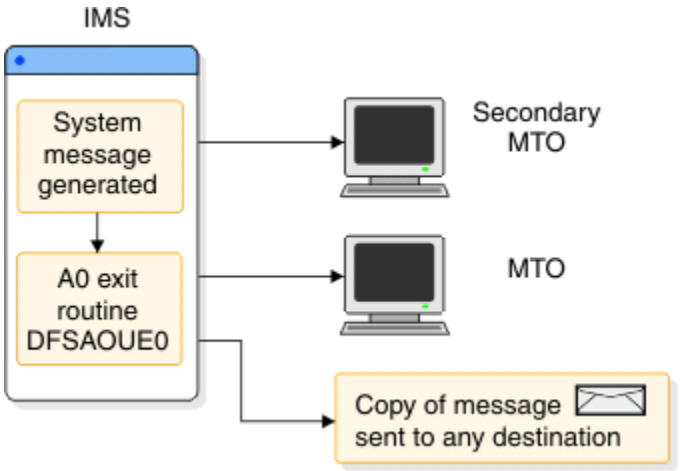


Figure 24. Processing when a system message is generated

A sample exit routine is described in Table 172 on page 439.

Activating this routine

DFSAOUE0 is activated after IMS restart is complete. DFSAOUE0 is activated for each system message, command, and command response.

When IMS shutdown processing begins, DFSAOUE0 is disabled and no longer receives control.

Types of messages passed to this routine

The following sections contain information about which messages are passed to the exit routine. IMS passes a copy of commands and command responses, and system messages destined for the master terminal.

A message that is passed to the exit routine can contain multiple segments.

System messages

A system message is a DFS message that is not a direct (synchronous) response to a command. IMS passes to the exit routine system messages that are destined for the master terminal. (If the system message is destined for the secondary master terminal or z/OS system console, IMS does not pass the exit routine a copy of the message.) Bit UEH1CPYP in the UEHFLG1 flag field of the UEHB (user exit header block) indicates that the input to the exit routine is a system message. In most cases, the first system message that causes IMS to call the exit routine is the DFS994 checkpoint message.

While IMS passes a system message destined for the master terminal, it can also send a copy of this message to the secondary master terminal or the z/OS system console. IMS sends this copy before it passes the original message to the master terminal. The exit routine can change only the original message that is destined for the master terminal. The copy that the secondary master terminal or z/OS system console receives does not reflect any changes the exit routine makes to the original message.

Most system messages are single-segment messages. Some system messages are multisegment messages (such as DFS802, DFS970, DFS2503, and DFS3222).

Commands

IMS passes the exit routine a copy of each IMS command that is entered except:

- Internally generated commands.
- Commands that are issued by a CMD or ICMD call from an AO application.
- `/FORMAT`
- `/LOOPTEST`
- `/MSVERIFY`
- `/RELEASE`
- `/NRESTART`
- `/ERESTART`

IMS passes the command after the message editing routines have been called to modify it. This modified input can contain carriage control characters.

Commands with network-qualified LU names

If you use network-qualified LU names at your installation, the LU name can be 17 bytes long. For IMS commands, the network-qualified LU names must be enclosed in single quotes (for example, 'NETID.LUNAME').

If an IMS command with the network-qualified LU name is passed to the AO exit routine, IMS modifies the network-qualified LU name in the input command before the command is passed to the AO exit routine. The single quotes around the network-qualified LU name are replaced with blanks, and the period that separates the network-identifier and the LU name is replaced with a colon.

Example: A /DISPLAY command with a network-qualified LU name entered at the terminal as:

```
/DISPLAY LUNAME 'NETWORK1.LUNAME1' LUNAME2 INPUT.
```

is passed to the AO exit routine or logged to the secondary master as:

```
/DISPLAY LUNAME  NETWORK1:LUNAME1  LUNAME2 INPUT.
```

Command responses

IMS passes a copy of command responses to the exit routine. A command response is a copy of the original response that IMS sent to the terminal that entered the command. Any asynchronous system message that IMS produces as a result of a command is **not** considered a command response, and is passed to the exit routine only if its destination is the master terminal (as is the case with all system messages that IMS passes to the exit routine). The exit routine can request that the edited command buffer be made available on the last entry by setting a flag in the UEHB (user exit header block).

To receive a command response, the exit routine must handle multisegment messages and check for subsequent segments; this is because the first segment of a command response is considered the second segment of a command, even if the response has only one segment. Responses to the /DISPLAY, /RDISPLAY and /RMxxxx commands are multiple segment responses.

Changes the command editor makes

The command editor translates certain control characters in any commands you enter from a terminal. You must accommodate this translation when you write your exit routine.

The translation is shown in the following table:

Table 173. Translation of control characters in commands

From	To
X'14' Restore	X'5D' Right parenthesis
X'15' New line	X'40' Blank
X'24' Bypass	X'4D' Left parenthesis
X'40' Blank	X'40' Blank
X'4B' Period	X'4B' Period
X'4D' Left parenthesis	X'4D' Left parenthesis
X'5D' Right parenthesis	X'5D' Right parenthesis
X'60' Dash	X'60' Dash
X'6B' Comma	X'6B' Comma
X'6D' Dash	X'40' Blank
X'7E' Equal	X'40' Blank

Types of messages not passed to this routine

IMS does not pass all system messages, operator-entered commands, and command responses to this exit routine. The following are messages that IMS does not pass to the exit routine:

- Messages resulting from a /BROADCAST command, other than the command and the initial response
- Messages that are associated with the /FORMAT, /LOOPTEST, /MSVERIFY, and /RELEASE commands and their responses
- Copies of system messages that are destined for the secondary master terminal or the z/OS system console
- Copies of message switches, messages that are inserted by application programs, or messages that result from the /BROADCAST command
- All system messages, commands, and command responses if message queues are unavailable, which is possible when initializing, restarting, or shutting down IMS

Single- and multisegment messages

The exit routine cannot determine from the first segment of a message whether it is a multisegment message or not. You can determine which messages are single-segment messages, and write the exit routine so that IMS only calls it once. This practice helps your system avoid additional processing that is incurred when the exit routine is written to always test for subsequent segments. To handle those messages that are multisegment messages or for which you cannot predetermine the number of segments, you can write the exit routine to request all remaining segments.

If you write the exit routine so that it does not differentiate between single and multisegment messages and always checks for remaining segments, IMS always calls it at least twice. A message segment can accompany the last entry to the exit routine when the message is a multisegment message. No segment is presented to the exit routine when the message is a single-segment message.

Supporting multisegment messages

Although most messages contain only one segment, some messages are multisegment messages. System messages DFS970 and the response to a /DISPLAY command are examples of multisegment messages. Even if a command response is a single-segment response, the exit routine must be written to handle multisegment messages; this is because a command always precedes a command response. If the exit routine does not check for subsequent segments, IMS does not pass the segments that contain the command response.

You can write the exit routine so that IMS calls it for each segment of a message. If you write the exit routine to request the remaining segments, the exit routine is called at least twice for each message, even if the message has only one segment. In this case, the last entry to the exit routine is not accompanied by a segment; this is because the message is a single segment. If the message is a multisegment message, the exit routine is called for the subsequent segments. The exit routine must test for a segment the last time it is entered.

For subsequent entries to the exit routine for a multisegment message, bit UEH1SEG is set in the UEHBFLG1 field of the UEHB (user exit header block) when another segment is being presented. UEHCPYBF points to the next segment of the message.

The exit routine cannot necessarily tell which segment belongs to which message because the presentation of segments that are associated with any one message can be interspersed with segments associated with a different message. The UEHB is unique for each message, and you can use the UEHURSVD field to track which message IMS presents to the exit routine.

Format of message segment copies

IMS uses the UEHB (user exit header block) to pass a copy of the message segment to the exit routine and places the address of that message segment in the UEHCPYBF field of the UEHB. The format of the copy of a system message, operator-entered command, or command response is shown in the following figure.

Figure 25. Message segment copy format

The message segment copy contains the following fields:

LL
2-byte field that contains the length of the message on first entry to the exit routine, not including the length of the 20-byte work area. (If your exit routine deletes or changes a message, or uses the work area, it must update this length field.)

ZZ
2-byte field reserved for IMS.

message text

System message
The first segment of the message text begins with the DFS*xxxx* number, indicating which message caused IMS to call the exit routine. The message number is followed by the text of the system message. If it is a multisegment message, the remaining segments contain additional text, but do not contain the DFS*xxxx* message number.

Command
The message text is one segment long and begins with the delimiter '/', followed by the command.

Command response
The command response is usually a DFS*xxxx* message or one segment of a multisegment command.

CR
Optional 1-byte field that contains carriage control characters (for example, X'15'). Input commands do not include a carriage control character. If the CR field is included, one byte is included in LL.

work area
A 20-byte work area added to the end of the system message, command, or command response; the exit routine can use this work area to communicate with the alternate destination for that segment.

Viewing the edited command buffer

IMS expands certain commands and places this expanded view into the edited command buffer. You can examine this buffer by setting the appropriate exit registers.

One occasion for examining the buffer is when command processing exceptions occur, indicated by the DFS058 XXX COMMAND COMPLETED EXCEPT message. When a LINE, LINK, NODE, or PTERM keyword is used with inclusive or range parameters, or when a LINE, LINK, PTERM, or SUBSYS keyword is used with the ALL parameter, IMS expands the command in the edited command buffer to include the actual resource names or numbers, except for the /BROADCAST command. The PTERM ALL keywords are only expanded for the /PSTOP, /PURGE, /RSTART, /START, /STOP, and /MONITOR commands. When a NODE, LTERM, or USER keyword is used with a generic parameter and exceptions occur, IMS expands the edited command buffer with up to 10 of the specific resource names that are invalid and that match the generic parameter.

Only parameter passwords are shown in the edited command buffer; command passwords are not shown.

The following figure shows the format of the edited command buffer.

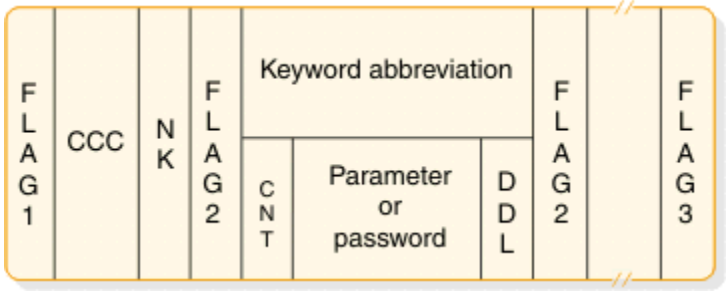


Figure 26. Edited command buffer

The fields contain the following information:

FLAG1
Field containing one of the following flags:

X'FE'
Beginning of the edited command.

X'FC'
An error was found in a parameter and this flag was set by the command action modules.

CCC
First 3 bytes of command.

NK
Hexadecimal value of the number of keywords in the edited command buffer.

FLAG2
Field containing one of the following flags:

X'FC'
Parameter that follows is in error.

X'FF'
3-byte keyword abbreviation follows.

X'FE'
Count (CNT) field and parameter follow.

X'('
Count (CNT) field and password follow.

Keyword Abbreviation
Refer to DFSCKWD0 to obtain the abbreviation. In some cases, the abbreviation is the first three characters of the keyword.

CNT
Number of characters in the parameter or password that immediately follow the CNT. This field is a 1-byte binary field.

Parameter or Password
Parameter exactly as entered from the terminal.

DDL
Delimiter that is entered after the parameter or password. If the ALL parameter is expanded to individual parameters, the delimiter is X'80'. If the parameter is generic, the delimiter is X'10'.

FLAG3
Period indicating the end of the command.

Restrictions

The exit routine can change or delete system messages only. It can modify the copy of the system message that the original destination (the master terminal) receives. It can also modify the copy that an alternate destination receives. The exit routine **cannot** change or delete the original command or command response. It can modify the copy of a command or command response that is destined for an alternate destination, but it cannot change the copy that the primary destination receives.

Some transactions must reside on the same IMS subsystem as DFSAOUE0 to process correctly. If your installation uses shared queues, define these local transactions as SERIAL to guarantee that they are processed on the local IMS subsystem. A transaction that is not defined as SERIAL can be processed on any IMS subsystem with a definition for the transaction.

Communicating with IMS

IMS communicates with this exit routine through the entry and exit registers, and the user exit header block (UEHB). IMS creates a UEHB for each message and passes it to the exit routine every time the exit routine is called for that message. Your exit routine can use the UEHURSVD field in the UEHB to store information about the message between each call to the exit routine for that message. The UEHB is freed and any values that were previously saved are lost when processing of the last segment of the message is complete.

Content of registers on entry

On entry, the exit routine must save all registers in the provided save area. The registers contain the following information:

Register	Contents
0	One of the following entry codes: **Entry Code** **Meaning** **0** First (initial) entry to the exit routine for the message. A segment is always presented to the exit routine (and the UEH1SEG field in the UEHB is set) with this entry code. The buffer pointed to by the UEHCPYBF field contains the first segment of the message that is being processed. The flag in the UEHBFLG1 field indicates what type of message it is. **4** Subsequent entry to the exit routine for the message. This entry code applies only to multisegment messages with three or more segments. The segment that is presented with this entry code is not the first or last segment. **8** Last entry to the exit routine for the message. This entry code applies only if the exit routine returned a return code of 0, 4, or 20 the last time it was called for this message (indicating that IMS continues to present the remaining segments to the exit routine). **12** Entry to exit routine after it requests storage. IMS passes the address of the buffer in the UEHUBUFF field. If storage is not available, UEHUBUFF contains 0 and the UEH1NSTG flag in the UEHBFLG1 field is set. The exit routine can attempt to get storage a second time, but if the second attempt is also unsuccessful, one of the following occurs: • If IMS is processing a command, it stops further processing of this command. • If IMS is processing a system message, it examines the size of the requested storage. If the size requested is greater than twice the value of UEHCPYSZ (the size of the current segment plus 20 bytes), IMS stops the exit routine for that message. If the size is equal to or less than this value, the exit routine waits for the storage to become available. **16** No message is presented to the exit routine. IMS stopped command processing because of errors in the command. IMS issues error messages, termination messages, or both. Command responses that were built and passed to the exit routine are canceled. A new response is built if the error is encountered while a / DISPLAY command response is being built.
1	Address of the UEHB.
7	Address of the communication terminal block (CTB).
9	Address of the communication line block (CLB) or partition specification table (PST).
11	Address of the system contents directory (SCD).
13	Address of the save area. The exit routine must not change the first 3 words. For external requests, the exit routine can chain down one save area to obtain the next available save area.
14	Return address to IMS.
15	Entry point of exit routine.

Key UEHB data fields on entry

The following are key UEHB entry fields:

UEHCPYBF
Address of the copy of the message IMS passed to the exit routine. If the UEH1SEG flag is set, the buffer contains a pointer to a copy of the message. If the UEH1CPYP flag in the UEHBFLG1 field is set, the buffer contains a copy of the first segment of a system message. If the UEH1CMD flag is set, the buffer contains a copy of the first segment of a command. If the UEH1CMD flag is set and the entry code is non-0, this field contains a copy of a segment of a command response.

UEHECMD
Address of the edited command buffer if this is the final entry to the exit routine and the UEH1ECMD flag in the UEHBFLG1 field was set (requesting the edited command buffer) the first time IMS called the exit routine.

UEHUBUFF
Address of an additional storage buffer, if the exit routine requests storage. If additional storage was not available, this field contains 0, and the UEH1NSTG flag in the UEHBFLG1 field is set.

Content of registers on exit

Before returning to IMS, the exit routine must restore all registers except for registers 0, 1, and 15, which contain the following:

Register	Contents
0	If register 15 contains a return code of 0 or 8 (and your exit routine sets the destination for the first time or changes it), this register contains the address of the alternate destination name. The alternate destination can be a transaction or an LTERM. The alternate destination name must be 8 bytes, left-justified with blanks. If the alternate destination is not a valid transaction or LTERM and the Extended Terminal Option (ETO) is set, a dynamic LTERM is created. (For more information on the ETO feature, see *IMS Version 15.2 Communications and Connections*.) If register 15 contains a return code of 16 (requesting additional storage), this register contains the size of storage requested for the user buffer. If the alternate destination was set with a previous return code of 0 (and your exit routine does not change it), this register contains 0.
1	If register 15 contains a return code of 0 or 8, this register contains the address of the segment to insert to the alternate destination or register 1 contains 0 to enqueue a previously inserted segment. If the segment to be inserted is the final segment, register 1 must contain the address of the message segment. If the segment is longer than the original segment (such as when your exit routine changes a message), and the device associated with the LTERM does not support the segment length, the terminal device can truncate the segment.
15	One of the following return codes:

Register	Contents	
	Return code	**Meaning**
	0	Insert the segment to the alternate destination and continue presenting the remainder of the segments to the exit routine.
	4	Do not insert the segment to an alternate destination. The exit routine can change the segment, or it can set the segment length to 0 to delete the segment. IMS continues to present remaining segments to the exit routine, which it can also change or delete. If register 0 contains 8 on entry, this return code causes all previously inserted segments to be enqueued.
	8	If register 1 contains 0, this return code instructs IMS to enqueue the **previously** inserted segments to the alternate destination (not to insert new segments). Processing is considered complete. If register 1 contains the address of the segment to insert, this return code instructs IMS to insert the current segment, if there is one, to the alternate destination and enqueue all inserted segments. Even if there are additional segments, this return code indicates that the remaining segments are **not** to be presented to the exit routine. Processing is considered complete.
	12	Cancel any segment already inserted to the alternate destination and indicate processing is complete.
	16	Request a user buffer in order to process this message (/ASSIGN command).
	20	Cancel all prior segments inserted to an alternate destination and continue presenting the remainder of the segments to the exit routine.

IMS checks to make sure that the return codes and alternate destination name are valid. If an invalid return code or an invalid alternate destination is returned, the exit routine is disabled for the remainder of the segments and is not called. IMS sends a trace record and a DFS2180I AUTOMATED OPERATOR USER EXIT ERROR - CODE=x message to the master terminal.

Key UEHB data and flag fields on exit

The following are key UEHB exit fields:

UEHCPYBF
Address of buffer that contains a copy of the segment going to the master terminal if a system message is being processed. If the exit routine changes the length of the segment in UEHCPYBF, the LL field must also be changed to reflect the new length. The LL field can be increased by up to 20 bytes or can be set to 0 (to delete the system message destined for the master terminal).

UEH1ECMD
Flag in the UEHBFLG1 field indicating that the exit routine requests the edited command buffer. The exit routine must set this flag the first time it is called.

UEHURSVD
The 20 bytes of storage reserved for the exit routine. The exit routine can use UEHURSVD to save the message being processed, entry codes, or flags between each invocation of the exit routine for a particular message.

Messages inserted to transactions by this routine

When you issue a GU call, and an AO application obtains a message that was inserted by the exit routine, the application I/O PCB contains an input LTERM name. IMS will determine the LTERM name as follows:

- If IMS calls the exit routine because of a system message, the input LTERM name is the master terminal name.
- If IMS calls the exit routine because of command input, the input LTERM name is the LTERM that entered the command.

Restriction: In a shared-queues environment, transactions that must be processed by the local IMS subsystem must be defined as SERIAL to process correctly. A transaction that is not defined as SERIAL can be processed on any IMS subsystem that has that transaction defined.

Related concepts

IMS Automated Operator Interface (AOI) (Operations and Automation)

Related reference

"Type-2 Automated Operator user exits (DFSAOE00 and other AOIE type exit routines)" on page 462
Type-2 Automated Operator user exits (DFSAOE00 and other AOIE type exit routines) are called continuously for system messages that are destined for the master terminal, for type-1 commands that are entered from terminals or from APPC or OTMA connections, and for the responses to those type-1 commands. The exit routine is also called for internal commands and commands from the ICMD call but not for their responses.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13
IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

AO functions and how to implement them

This example shows how to perform each of the AO functions on all segments of a multisegment message.

You can write this exit routine to perform a number of functions. You can use this example as a guideline for writing your own exit routine. You can write the exit routine to support only single-segment messages. This example requests a user buffer for some of the functions in which to store a copy of the message segment. You can use a different storage area to store a copy of the message segment.

Related reference

"Setting up the exit registers" on page 454
Set up exit registers to perform and support certain functions for single-segment and multisegment messages.

Ignore selected segments or an entire message

If the exit routine is not interested in the message segment, it can set the exit registers to ignore them and resume processing.

The exit routine is called for system messages destined for the master terminal, operator-entered commands, and command responses regardless of whether the exit routine is interested in the message. The segment is ignored by setting the following register:

Register 0 on entry	Registers on exit
0	Register 15 = 12

Send copy of message to alternate destination

You can write the exit routine so that IMS sends a copy of a system message destined for the master terminal, an operator-entered command, or a command response to an alternate destination in addition to the original destination.

For the first entry to the exit routine, insert the segment to the alternate destination and request remaining segments, if there are any, by setting the following register:

Register 0 on entry	Registers on exit
0	Register 0 = address of alternate destination name Register 1 = address of message (UEHCPYBF) Register 15 = 0

For subsequent entries that are not the last entry, insert the segment to the alternate destination and request remaining segments by setting the following register:

Register 0 on entry	Registers on exit
4	Register 0 = address of alternate destination name Register 1 = address of message (UEHCPYBF) Register 15 = 0

For the last entry to the exit routine, insert the segment to the alternate destination, enqueue all of the segments, and indicate that processing is complete by setting the following register:

Register 0 on entry	Registers on exit
8	Register 0 = address of alternate destination name Register 1 = address of message (UEHCPYBF) Register 15 = 8

Send new message to alternate destination

You can write the exit routine to send a new message to an alternate destination for each system message, operator-entered command, or command response that is passed to the exit routine. The original message proceeds to its destination unchanged, and a completely new message is sent to an alternate destination.

For the first entry to the exit routine for this message, the exit routine must request the storage in which to build each segment of the new message. The buffer requested during this initial entry must be large enough to fit the largest message segment you plan to send. The exit routine cannot request additional storage during subsequent entries for this message. Request enough storage for a message segment by setting the following register:

Register 0 on entry	Registers on exit
0	Register 0 = size of message segment Register 15 = 16

For the next entry to the exit routine after successfully getting storage for this message segment, move the first segment of the new message to the user buffer (UEHUBUFF), and set the message length in the first 2 bytes. Insert the message segment to the alternate destination and request remaining segments, if any, by setting the following register:

Register 0 on entry	Registers on exit
12	Register 0 = address of alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 0

For subsequent entries that are not the last entry, move the next segment of the new message into the user buffer. The user buffer is reused for each segment of the message. Set the message length in the first 2 bytes of the user buffer. Insert the message segment to the alternate destination and request the remaining message segments by setting the following register:

Register 0 on entry	Registers on exit
4	Register 0 = address of alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 0

For the last entry to the exit routine for this message, move the last segment of the new message into the user buffer. Set the message length in the first 2 bytes. Insert the segment to the alternate destination, and indicate that processing is complete by setting the following register:

Register 0 on entry	Registers on exit
8	Register 0 = address of alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 8

Change system message text

You can change the original text of a system message destined for the master terminal.

The system message that is passed to the exit routine includes 20 bytes that are added to the end of the message. The exit routine can use these 20 bytes. Changes to the system message are limited to the original message length, plus 20 bytes. If the changed message includes the 20-byte area provided at the end, the exit routine must increment the message length field by 20.

For each entry to the exit routine for this message, change the system message text. For the first entry to the exit routine, allow the changed segment to proceed to its master terminal destination and request remaining message segments by setting the following register:

Register 0 on entry	Registers on exit
0 or 8	Register 15 = 4

Related reference

“Change message text and send to alternate destination” on page 451
The exit routine can change the copy of a system message destined for the master terminal and send the changed message to both the master terminal and an alternate destination.

Change message text and send to alternate destination

The exit routine can change the copy of a system message destined for the master terminal and send the changed message to both the master terminal and an alternate destination.

If the copies sent to the master terminal and the alternate destination are different, your exit routine needs to request storage for the user buffer for the copy sent to the alternate destination. The exit routine cannot change the copy of the command or command response that is in the copy buffer.

The copy of the message passed to the exit routine has an additional 20 bytes added to the end, which the exit routine can use. Changes to the message are limited to the original message length, plus this 20 bytes. If the changed message includes the 20-byte area, the exit routine must increment the message length field by 20.

For the first entry to the exit routine for this message, change the message text. If the changed message includes the 20-byte area provided at the end, increment the message length field by 20. Insert the segment to an alternate destination, and request the remaining segments by setting the following register:

Register 0 on entry	Registers on exit
0	Register 0 = address of the alternate destination name Register 1 = address of the message (UEHCPYBF) Register 15 = 0

For subsequent entries that are not the last entry, change the message segment text. If the changed message includes the 20-byte area provided at the end, increment the message length field by 20. Insert the message segment, and send it to an alternate destination by setting the following register:

Register 0 on entry	Registers on exit
4	Register 0 = address of the alternate destination name Register 1 = address of the message (UEHCPYBF) Register 15 = 0

For the last entry to the exit routine for this message, change the message text. If the changed message includes the 20-byte area provided at the end, increment the message length field by 20. Insert the segment to an alternate destination, enqueue all of the segments, and indicate that processing is complete by setting the following register:

Register 0 on entry	Registers on exit
8	Register 0 = address of the alternate destination name Register 1 = address of the message (UEHCPYBF) Register 15 = 8

Delete system message to MTO

Your exit routine can delete a system message segment that is destined for the master terminal. Delete a segment by setting the length field in the message buffer (the first two bytes) to 0. Your exit routine cannot delete commands and command responses.

For the first entry to the exit routine for the message, set the length field of the message to 0, and obtain the second segment by setting the following register:

Register 0 on entry	Registers on exit
0	Register 15 = 4

For subsequent entries that are not the last entry, set the length field of the message to 0, and obtain the next segment by setting the following register:

Register 0 on entry	Registers on exit
4	Register 15 = 4

For the final entry to the exit routine for the message, set the length field of the message to 0, and indicate that processing is complete by setting the following register:

Register 0 on entry	Registers on exit
8	Register 15 = 12

Delete system message to MTO and send copy to alternate destination

Your exit routine can delete a system message destined for the master terminal and send a copy to an alternate destination instead. It cannot delete commands and command responses.

Before deleting the system message, the exit routine must request storage for a user buffer in which to put a second copy of the message. Your exit routine must request enough storage to fit the largest segment of the message.

For the first entry to the exit routine for the system message, your exit routine can request storage by setting the following register:

Register 0 on entry	Registers on exit
0	Register 0 = size of the largest message segment Register 15 = 16

For the next entry after successfully getting storage for the largest message segment, move the first segment of the message from UEHCPYBF into the user buffer (UEHUBUFF), including the length in the first 2 bytes. Delete the message destined for the master terminal by setting the length field of the message segment pointed to by UEHCPYBF to 0. Insert the message copy to the alternate destination, and request the next segment by setting the following register on exit:

Register 0 on entry	Registers on exit
12	Register 0 = address of the alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 0

For the last entry to the exit routine for this message, move the last segment of the message into the user buffer. The user buffer is reused for each segment of the message. Delete the last segment destined for the master terminal by setting the length field of the message segment pointed to by UEHCPYBF to 0. Insert the last segment, enqueue the entire message, and indicate that processing is complete by setting the following register on exit:

Register 0 on entry	Registers on exit
8	Register 0 = address of alternate destination name Register 1 = address of the message segment (UEHUBUFF) Register 15 = 8

Request the edited command buffer

Your exit routine can request the edited buffer that was created for an input command.

On first entry, request the edited command buffer by setting flag UEH1ECMD on in the UEHBFLG1 field. Request the next command response segment by setting the following register on exit:

Register 0 on entry	Registers on exit
0	Register 15 = 4

For subsequent entries that are not the last entry to the exit routine for this command response, continue requesting the next command response segment by setting the following register on exit:

Register 0 on entry	Registers on exit
4	Register 15 = 4

For the last entry to the exit routine for this command response message, the UEHECMD field contains the address of the edited command buffer. If the edited command buffer is not available (such as when there are command syntax errors), the UEH1CBNA flag is set in the UEHBFLG1 field, and the UEHECMD field contains 0.

Setting up the exit registers

Set up exit registers to perform and support certain functions for single-segment and multisegment messages.

The following tables describe how to set up exit registers to perform certain functions for single-segment and multisegment messages. Refer to both tables if you are writing your exit routine to support single-segment and multisegment messages. If you can identify which messages are single-segment messages and which are multisegment messages, you can write the exit routine to handle each type differently.

Subsections:

- "Single-segment messages" on page 454
- "Multisegment messages" on page 455

Single-segment messages

The following table shows how to set up registers on exit for single-segment messages. If your exit routine only examines single-segment messages, or if you can identify which messages are single-segment messages (and can use this logic), you can use this information to write your exit routine.

Table 174. Exit functions for single-segment messages

Function	Register 0 on entry	UEHCPYBF length field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Ignore entire message	0				12
Send copy of message segment to alternate destination	0		Address of alternate destination name	Address of message (UEHCPYBF)	8
Send new message to alternate destination	0		Size of message		16
	12		Address of alternate destination name	Address of message (UEHUBUFF)	8
Change system message	0	Length + 20			8
Change message segment and send to alternate destination	0	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	8
Delete system message to master terminal	0	0			8

Table 174. Exit functions for single-segment messages (continued)

Function	Register 0 on entry	UEHCPYBF length field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Delete system message to master terminal and send copy to alternate destination	0		Size of message		16
	12	0	Address of alternate destination name	Address of message (UEHUBUFF)	8

Multisegment messages

The following table shows how to set up the registers on exit for multisegment messages. If your exit routine examines multisegment messages, or if you can identify which messages are multisegment messages (and can use this logic), you can use the information in this figure to write your exit routine. All values given are in decimal format.

Table 175. Exit functions for multisegment messages

Function	Register 0 on entry	UEHCPYBF length field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Ignore entire message	0				12
Send copy of message to alternate destination for each segment	0		Address of alternate destination name	Address of message (UEHCPYBF)	0
	4		Address of alternate destination name	Address of message (UEHCPYBF)	0
	8		Address of alternate destination name	Address of message (UEHCPYBF)	8
Send new message to alternate destination for each segment	0		Size of storage to get for largest message segment		16
	12		Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	0
	4		Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	0
	8		Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	8

Table 175. Exit functions for multisegment messages (continued)

Function	Register 0 on entry	UEHCPYBF length field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Change each segment of system message	0	Length + 20			4
	8	Length + 20			4
Change each segment of a message and send to alternate destination	0	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	0
	4	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	0
	8	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	8
Delete each segment of system message to master terminal	0	0			4
	8	0			12
Delete each system segment to master terminal and send copy to alternate destination	0		Size of storage to get for largest message segment		16
	12	0	Address of alternate destination name	Address of message segment (UEHUBUFF)	0
	8	0	Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	8
Request the edited command buffer	0				4
	4				4
	8				12

Related reference

"AO functions and how to implement them" on page 449
This example shows how to perform each of the AO functions on all segments of a multisegment message.

User Exit Header Block (UEHB)

The UEHB contains three categories of data and flag fields.

The UEHB contains the following data and flag fields. The following table indicates the field name, length in bytes, and description of the data fields, and it indicates the field name, hexadecimal value, and meaning of the flag fields.

Data and flag fields in the UEHB can be grouped into one of three categories, depending on how the exit routine can use them.

Modifiable
The exit routine can change these fields to communicate with IMS or to use as a work field.

Read only
The exit routine can read but not modify these fields.

Reserved
The exit routine cannot use these fields. They are reserved for use by IMS.

Table 176. UEHB field descriptions

Field	Length/Value	Description
UEHSRCE	4 bytes	Address of source CNT. Usage = read only. This field points to the source LTERM of the message segment. For a system message, the source is the master LTERM. For a command, the source is the LTERM where the command was entered.
UEHDEST	4 bytes	Address of destination CNT. Usage = read only. This field points to the destination of message segment. This is the destination of the 'presented' message segment, not the alternate destination that the exit routine can define.
UEHUBUFF	4 bytes.	Address of user buffer. Usage = read only. The buffer pointed to by this field is acquired when the exit routine returns a return code of 16 in register 15. The buffer can contain a copy of a message to be inserted to an alternate destination. If the buffer contains a copy of a message, the exit routine must update the 2-byte length field. If there is no message in the buffer, the length field is not necessary.
UEHCPYBF	4 bytes	Address of exit routine copy buffer. Usage = read only. The buffer pointed to by this field contains a copy of the system message segment, command, or command response that IMS passes to the exit routine. This area is the size of the message segment + 20 bytes (for modification). The first two bytes are the length field. For each entry to the exit routine, IMS reuses the copy buffer if the buffer created for a prior call is greater than or equal to the size that the current call requires. Otherwise, IMS frees the prior buffer and creates a new one.
UEHECMD	4 bytes	Address of edited command buffer. Usage = read only. On last entry to the exit routine, this field contains the address of the edited command buffer if the UEH1ECMD flag in the UEHBFLG1 field was set during the first entry to the exit routine for the message.
UEHIPCB	4 bytes	Address of input PCB. Usage = reserved.

Table 176. UEHB field descriptions (continued)

Field	Length/Value	Description
UEHIWRK1	4 bytes	Internal work area. Usage = reserved.
UEHIWRK2	4 bytes	Internal work area. Usage = reserved.
UEHIWRK3	4 bytes	Internal work area. Usage = reserved.
UEHBMODN	8 bytes	MFS MOD name. Usage = modifiable. The MFS MOD name sent with the message to the alternate destination that the exit routine specifies.
UEHPOPCB	28 bytes	Alternate PCB. Usage = reserved. Used for either AO transaction or any other transaction or LTERM destination that the AO specifies.
UEHSCPCB	28 bytes	Secondary master PCB. Usage = reserved.
UEHCPYSZ	2 bytes	Size of UEHCPYBF copy buffer. Usage = reserved.
UEHNODE	8 bytes	Nodename. Usage = read only. VTAM nodename or 0. If a command was entered, this is the nodename of the VTAM terminal that entered the command. If a system message is being passed and the master terminal is a VTAM node, this is the nodename of the VTAM master terminal node.
UEHHSQN	8 bytes	User name. Usage = read only. User name of user signed on to node or the Intersystem Communication (ISC) user associated with the node that entered the command, if UEH1CMD is set on in UEHBFLG1. User name or 0.
UEHOCALL	2 bytes	Usage = reserved.
UEHBFLG1	1 byte	Flag byte 1 for AOI and exit routine as follows:
UEH1ECMD	X'80'	Indicates that the exit routine requests the edited command buffer. Usage = modifiable. If the exit routine sets this flag on the first entry, the UEHECMD field points to the edited command buffer on the last entry. Also see UEH1CBNA flag.

Table 176. UEHB field descriptions (continued)

Field	Length/Value	Description
UEH1SEG	X'40'	Indicates that a segment is presented to the exit routine. Usage = read only. This flag is set when a segment is present in the UEHCPYBF field for the exit routine's examination. It is reset if entry code = 8 and prior call was PUT MOVE (segment already presented to exit routine on previous call with entry code = 0 or 4).
UEH1CPYP	X'20'	Indicates that the exit routine was called for a system message destined for the master terminal. Usage = read only. This flag is set when an asynchronous system message caused IMS to call the exit routine. A UEHB is created and the message is present in the UEHCPYBF field. The copy buffer contains the message plus 20 bytes (which the exit routine can modify).
UEH1CMD	X'10'	Indicates that the exit routine was called for a command or command response (if the entry code is non-0). Usage = read only. This flag is set when a command caused IMS to call the exit routine. A UEHB (user exit header block) is created and the command is present in the UEHCPYBF field. This flag is set until command processing is completed.
UEH1NSTG	X'08'	Storage not available for the user buffer. Usage = read only. Set when a conditional request for storage for the user buffer cannot be satisfied.
UEH1CBNA	X'04'	Edited command buffer not available Usage = read only. Set if the edited command buffer was not constructed by the command processor due to errors.
UEH1PSTD	X'02'	PST dispatch. Usage = reserved. System message being issued as a result of application program processing.
UEHBFLG2	1 byte	Flag byte 2 for AOI.
UEH2BYP	X'80'	Exit routine does not want rest of message. Usage = reserved. Set when the exit routine returns a return code of 8 or 12 indicating no more message segments are to be presented.

Table 176. UEHB field descriptions (continued)

Field	Length/Value	Description
UEH2POTR	X'40'	Alternate destination found. Usage = reserved. Set after the alternate destination is successfully found, when register 15 contains a 0 return code and an 8-byte alternate destination name is in register 0. The alternate destination can be a transaction or LTERM.
UEH2ILOC	X'10'	INSERT LOCATE was last call. Usage= reserved. Indicates that the prior segment is to be presented to the exit routine.
UEH2UENT	X'08'	Exit routine was entered at least once. Usage = reserved. Used to determine if entry code 0 is to be set.
UEH2LAST	X'04'	Set when entry code 8 is set on entry. Usage = reserved. Used to indicate that the exit routine was entered for the last time.
UEH2NCUR	X'02'	Set when current call is not yet processed. Usage = reserved. Used when entered for a final call but the exit routine had not yet been entered for the first time.
UEH2QNOP	X'01'	Set when call is not to be passed to the queue manager (QMGR). Usage = reserved. Used to indicate that a PUT MOVE should not be done if the exit routine is deleting a system message to the primary master terminal.
UEHBFLG3	1 byte	Flag byte 3 for AOI.
UEH3ILOC	X'80'	Current call is INSERT LOCATE. Usage = reserved.
UEH3PUTM	X'40'	Current call is PUT MOVE. Usage = reserved.
UEH3CANO	X'20'	Current call is CANCEL OUTPUT. Usage = reserved.
UEH3ENQ	X'10'	Current call is ENQUEUE. Usage = reserved.
UEH3TERM	X'08'	Current call is AOI TERMINATION. Usage = reserved.
UEH3VSEG	X'04'	Segment exists for M/T. Usage = reserved.
UEHBFLG4	1 byte	Error flag byte.

Table 176. UEHB field descriptions (continued)

Field	Length/Value	Description
UEH4ERRM	X'80'	AOI error message in progress. Usage = reserved.
UEH4SMER	X'40'	Secondary master terminal error. Usage = reserved.
UEH4FAIL	X'20'	Current call will be failed. Usage = reserved.
UEH4UEHB	X'10'	Previous UEHB exists. Usage = reserved.
UEHBERRC	1 byte	QAOI error code as follows:
UEHBERR1	C'1'	Invalid alternate destination. Usage = reserved.
UEHBERR2	C'2'	Queue manager return code. Usage = reserved.
UEHBERR3	C'3'	Invalid exit routine return code. Usage = reserved.
UEHBERR4	C'4'	Multiple user buffer request. Usage = reserved.
UEHBERR5	C'5'	User buffer storage not available. Usage = reserved.
UEHBERR6	C'6'	Previous UEHB exists. Usage = reserved.
UEHBERR7	C'7'	Usage= reserved.
UEHBFLG5	1 byte	Flag byte 5 for AOI.
UEH5LTRM	X'80'	Usage = reserved. Dynamic LTERM marked in use.
UEHIRSVD	2 bytes	Usage = reserved.
UEHURSVD	20 bytes	Work area reserved for exit routine. Usage = modifiable. The exit routine can use this field to keep track of message numbers, entry codes, and flags between invocations of the exit routine for a particular message. UEHURSVD can be used to tie segments of a multisegment message together, since remaining segments do not contain the message number.

Type-2 Automated Operator user exits (DFSAOE00 and other AOIE type exit routines)

Type-2 Automated Operator user exits (DFSAOE00 and other AOIE type exit routines) are called continuously for system messages that are destined for the master terminal, for type-1 commands that are entered from terminals or from APPC or OTMA connections, and for the responses to those type-1 commands. The exit routine is also called for internal commands and commands from the ICMD call but not for their responses.

The AO exit routine intercepts messages before IMS sends the system message, executes the terminal command, or sends the terminal command response. The AOIE type exit routine is also called for system messages destined to the secondary master if it was specified during IMS initialization.

You can write two types of Automated Operator (AO) exit routines. The AO exit routine described in this topic (DFSAOE00 and other AOIE type exit routines) is called a type-2 AO exit routine. It can be used in the DB/DC, DCCTL, and DBCTL environments.

The other AO exit routine (DFSAOUE0) is called a type 1 and can be used in the DB/DC and DCCTL environments.

If both DFSAOUE0 and an AOIE type exit routine are provided in a DB/DC or DCCTL environment, the AOIE type exit routine is called first. The AOIE type exit routine determines which exit routine processes the message, or command.

This exit is called twice: once for the secondary master system message and then for the primary master message.

Subsections:

- "About this routine" on page 462
- "Restrictions" on page 464
- "Communicating with IMS" on page 465

About this routine

IMS calls the AOIE type exit routine:

- Before messages are logged to the secondary master terminal (if the AOIE type exit routine indicated this during the initialization call)
- For IMS system messages destined for the master terminal (the z/OS system console in the DBCTL environment)
- For commands entered from a terminal and the responses to those commands
- For commands issued from an AO application using the DL/I ICMD call
- For commands generated internally by IMS

The AOIE type exit routine intercepts these communications before IMS sends the system message, executes the command, or sends the command response.

With the AOIE type exit routine, you can:

- Prevent system messages from being logged at the IMS secondary master terminal only if indicated during IMS initialization call.
- Modify the text of IMS system messages. The AOIE type exit routine can also add up to 20 bytes of additional text to the end of a message.

  Any synchronous system "messages" generated by IMS after a command has been entered (such as DFS058 cccc COMMAND IN PROGRESS) are not really messages; they are command responses and cannot be modified.
- Delete IMS system messages. In a DBCTL environment, no trace of the original IMS message to the z/OS system console is kept if the message is deleted or modified.

- Direct any message, command, or command response to an AO application.
- Start a BMP job (for example, an AO application). The AOIE type exit routine can issue SVC 34 to start a BMP. The /START command can override the APARM value on the EXEC statement. The APARM parameter is a way of passing information to the AO application. To retrieve the value specified on the APARM parameter, the AO application can issue the DL/I INQY call.

The following table shows the attributes for the type-2 AO exit routine.

Table 177. Automated operator user exit attributes (type AOIE exit routine)

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	You can name this exit routine DFSAOE00 and link it into a library that is included in the STEPLIB concatenation. Alternatively, you can define one or more exit routine modules with the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set. The routines are called in the order that they are listed in the parameter. If DFSAOE00 is linked into a library in the STEPLIB concatenation and the USER_EXITS section of the DFSDFxxx member defines exit routines, the exit routines defined in the DFSDFxxx member will be loaded. DFSAOE00 is only loaded if it is listed as one of the exit routines in the DFSDFxxx member.
Binding	This exit routine must be reentrant. You must manually link edit the routine with DFSCSI00 to include the routine.
Including the routine	DFSAOE00 is a stand-alone, 31-bit reentrant module that you provide. You specify DFSAOE00 by linking it with DFSCSI00 in the IMS.SDFSRESL concatenation as a stand-alone module. DFSAOE00 can then be loaded and called. If you link both DFSAOE00 and DFSAOUE0 (the other AO exit routine), both are loaded. DFSAOE00 is called first and can either process the message, or command, or it can return a code indicating DFSAOUE0 should do the processing instead. The module or modules must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation. No additional steps are necessary to use a single exit routine that is named DFSAOE00. If you use multiple exit routines or a single exit routine that is not named DFSAOE00, specify EXITDEF=(TYPE=AOIE,EXIT=(exit_names)) in the EXITDEF parameter of the USER_EXITS section of the DFSDFxxx member of the IMS.PROCLIB data set.
IMS callable services	The AOIE type exit routine can use callable services for storage and AOI functions. It is defined to IMS as a standard exit and receives the callable services token in the standard exit parameter list. This routine does not need to issue an initialization call (DFSCSII0).
Sample routine location	IMS.ADFSSMPL

Using AOI callable services

The AOIE type exit routine can use IMS AOI callable services to communicate with an AO application. Using AOI Services, AOIE type exit routine can pass a message containing one or more message segments to one or more AO applications. AOI callable services functions include:

- INSERT, which inserts a message segment to a message buffer.
- ENQUEUE, which sends a message to one or more AO applications using AOI token names. The message sent on the ENQUEUE request can be a single-segment or multisegment message built with INSERT requests or a single-segment message supplied on the ENQUEUE request. An AO application

issues a GMSG call, specifying an AOI token name, to retrieve a message sent from the AOIE type exit routine.

- CANCEL, which removes inserted segments when you decide not to send the message to an AO application.

Activating this routine

The AOIE type exit routine is activated:

- During IMS initialization. The purpose of this first entry is so that the AOIE type exit routine can, if you want, do its own initialization.
- After IMS restart is complete. The AOIE type exit routine is activated for each system message, command, and command response.

After IMS shutdown processing has begun, the AOIE type exit routine is disabled and no longer receives control.

AOIE type exit routine and the REFRESH USEREXIT command

AOIE type exit routines can be refreshed dynamically by using the REFRESH USEREXIT command if the exit routines are defined to IMS by using an EXITDEF statement in the USER_EXITS section of the DFSDFxxx member. The REFRESH USEREXIT TYPE(AOIE) command brings in new copies of the AOIE type exit routines. If AOIE type exit routines are being called for multi-segment output at the same time that the AOIE exit routines are being refreshed, the unchanged set of AOIE exit routines is called for each segment in the output. The new set of user exit routines is called for any subsequent system messages, commands, or command responses.

For example, if a /DIS ACTIVE command is entered at the same time as a REFRESH USEREXIT TYPE(AOIE) command, IMS continues to call the unchanged set of AOIE user exit routines for every segment in the command output that is called for the /DIS ACTIVE command input. A subsequent /DIS ACTIVE command and its responses are called with the new set of exits. If the REFRESH USEREXIT TYPE(AOIE) is in the process of deleting a set of AOIE user exit routines in IMS, the unchanged set of AOIE exit routines is called until the command sequence is complete. After the deletion of all AOIE user exit routines is complete, subsequent command and system messages are not passed to any AOIE type exit routines.

If an AOIE type exit routine was not defined to IMS and the REFRESH USEREXIT TYPE(AOIE) command adds the AOIE type exit routine, IMS calls the exit routine with the INIT function before the exit routine is called for any messages or commands.

Multiple AOIE type exit routines

When an AOIE type exit routine is called as part of a list of AOIE type exits, each AOIE type exit routine is called with the same parameter list. IMS does not refresh the parameter list or use information from the parameter list between calls to the individual AOIE exit routines. If your exit sets the return code parameter (AOEORPLY) to anything other than zero (AOE0IGNR), your exit routine must set the SXPLCNXT exit parameter to SXPL_CALLNXTN so that IMS does not call any more of the AOIE type exit routines in the list. Otherwise, a subsequent AOIE type exit could modify AOE0RPLY and overwrite your modification.

For example, if an AOIE type exit needs to add text to a message, it should complete the following steps:

- Add the text to the end of the message.
- Set the return code parameter (AOEORPLY) to AOE0MODF.
- Set the SXPLCNXT exit parameter to SXPL_CALLNXTN.

Refreshes are performed at the exit type level, so when one AOIE type exit routine is modified, all AOIE type exit routines are refreshed.

Restrictions

All messages queued to an AO application by an AOIE type exit routine remain in the IMS subsystem in which they are queued. Only AO applications in the local subsystem can issue a DL/I GMSG call to

access the queued messages. Because these messages are not written to a shared-queues structure, applications on other IMS subsystems cannot access them.

You cannot use an AOIE type exit routine to modify or delete commands and command responses. This includes commands from a terminal or an application program, or internally generated commands.

Communicating with IMS

IMS communicates with an AOIE type exit routine through the entry registers and a parameter list.

Content of registers on entry

The content of the registers that are passed from IMS to this exit routine each time it is activated follows:

Register	Content
1	Address of the "IMS standard user exit parameter list" on page 5
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is not chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard exit parameter list

This exit routine uses the Version 6 standard exit parameter list. The address of the work area passed to this exit routine in SXPLAWRK will be the same each time that this exit routine is called.

Function-specific parameter list

The following table shows the contents of the function-specific parameter list. The address of this parameter list is in the standard exit parameter list field SXPLFSPL.

Table 178. Function-specific parameter list for the AOIE type exit routine (mapped by DFSAOE0)

Field	Offset	Length	Description
AOE0VER	0	4	Address of word containing version number for DFSAOE0.
AOE0FUNC	4	4	Reason for entering AOIE: **1** Initial entry. The AOIE type exit routine can do initialization functions. **2** Message segment to process. **3** Command is aborted. **4** A message segment for the secondary master terminal is passed to the AOIE type exit routine. The AOIE type exit routine can return to IMS with AOE0RPLY=3 (AOE0CNCL) to prevent the message from being enqueued to the secondary master. AOE0RPLY=0 (AOE0IGNR) will allow the message to be queued to secondary master. Any other response value results in message DFS2180, the response is ignored, and the message is queued to the secondary master.
AOE0SEG	8	4	Address of message buffer or 0 if this is initial entry. (The next table shows the message buffer.)

Table 178. Function-specific parameter list for the AOIE type exit routine (mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0WRKA	12	4	Address of 256-byte work area used by the AOIE type exit routine. The area is static for the segments of a message, or for a command and the related command responses. **Note:** The same work area is passed to all AOIE exits if more than one AOIE exit module is defined in the system. For more details, see section "Available work areas for the AOIE user exit" at the end of this topic.
AOE0FLG1	16	1	Entry codes: **X'80'** First segment of multiple segments or first and only segment when X'20' is also on. **X'40'** Middle segment of multiple segments. **X'20'** Last or only segment. **X'10'** Command response will be sent for this command. X'20' is also set when X'10' is set. **X'08'** No segment presented. Last entry to exit.
AOE0FLG2	17	1	Segment or command type: **X'80'** Command entered at terminal. **X'40'** Command response segment. **X'20'** Command (ICMD) issued by AO application. **X'10'** Command generated internally by IMS. **X'08'** IMS system message segment.
AOE0FLG3	18	1	**X'80'** Command input entered at a terminal exceeded 256 bytes.
AOE0USII	19	1	Indicator for contents of user ID field: **U** user ID **L** LTERM **P** PSB name **O** Other name

Table 178. Function-specific parameter list for the AOIE type exit routine (mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0DMTK	20	4	Directed message token required to issue AOI callable service requests.
AOE0IMSI	24	8	IMS subsystem identifier.
AOE0IMSL	32	4	IMS version and release.
AOE0SSTY	36	1	**X'01'** DB/DC system **X'02'** DCCTL system **X'03'** DBCTL system
AOE0ROLE	37	1	**X'01'** XRF active **X'02'** XRF alternate IMS **X'05'** FDBR region
AOE0MVSL	38	1	z/OS version and release on which IMS was generated.
AOE0ENVR	39	1	AO exit routine environment: **X'1'** DFSAOUE0 is loaded. Commands and messages can be passed to DFSAOUE0 to process.
AOE0ORGC	40	4	Origin of the command: **0** Origin fields not set. Fields are set for commands entered from a terminal, an MCS console, LU 6.2 conversation, or OTMA client in a DB/DC or DCCTL system. **1** Origin other than that defined by a specific code **2** VTAM terminal **3** LU 6.2 conversation **4** MCS/E-MCS console **5** OTMA client **6** System console **7** Master terminal
AOE0LINE	44	4	Terminal line number (AOE0ORGC=1).
AOE0NODE	44	8	VTAM node name (AOE0ORGC=2).

Table 178. Function-specific parameter list for the AOIE type exit routine (mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0NWID	44	8	Network ID (AOE0ORGC=3).
AOE0CONS	44	4	The 4-byte MSC/E-MSC terminal ID (AOE0ORGC=4).
AOE0TMEM	44	16	OTMA member name (AOE0ORGC=5).
AOE0PTRM	48	4	Physical terminal number (AOE0ORGC=1).
AOE0MCSU	48	8	User identification (AOE0ORGC=4).
AOE0LTRM	52	8	Logical terminal name or blanks if LTERM does not exist (AOE0ORGC=1,2).
AOE0LUNM	52	8	Logical unit name (AOE0ORGC=3).
AOE0USID	60	8	Signed-on user ID or blanks (AOE0ORGC=1,2).
AOE0LUUS	60	8	User ID (AOE0ORGC=3).
AOE0TPIP	60	8	Pipe (AOE0ORGC=5).
AOE0USER	68	8	VTAM user, subpool name, or blanks (AOE0ORGC=2).
AOE0TUSR	68	8	OTMA user ID (AOE0ORGC=5).

Table 178. Function-specific parameter list for the AOIE type exit routine (mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0RPLY	76	4	Return code from the AOIE type exit routine. This is the only field in this parameter list that the AOIE type exit routine can modify. **0** AOIE is not interested in this message or command segment. IMS will process the message as if the user exit did not exist. Subsequent segments of the message or command response are presented to AOIE. **1** AOIE is not interested in this message or command segment. Call DFSAOUE0 for processing. Do not call AOIE for subsequent segments of this message. **2** Send no more segments for this message or command to AOIE. **3** Delete the IMS system message segment. **4** Delete the IMS system message segment and all subsequent segments of this message. **5** AOIE modified the IMS system message segment. **6** Tells IMS to call the AOIE exit routine for secondary master messages. If secondary master logging is in effect, AOIE gets called twice for each segment: first for the secondary master message, and second for the AOI exit processing. This return code can be returned only on the initial entry to AOIE (AOE0FUNC=1). If it is returned for any other function call, message DFS2180 is issued, indicating the error, and the response is treated as AOE0RPLY=0.

Message buffer

The message buffer contains an IMS message, command, or command response. The following table shows the format of the message buffer. The function-specific parameter list points to this buffer.

Table 179. Message buffer

Field	Length	Description
LL	2	Length of the message on first entry to exit routine. Length includes the length of the LL and ZZ fields but excludes length of the 20-byte work area. (If your exit routine deletes or changes a message or uses the work area, it must update this field.)
ZZ	2	Zeros.

Table 179. Message buffer (continued)

Field	Length	Description
Message text	Variable	Text of the message, command, or command response. **System message:** The first segment of the message text begins with the DFSxxxx number, indicating which message caused IMS to call the exit routine. The message number is followed by the text of the system message. If it is a multisegment message, the remaining segments contain additional text, but do not contain the DFSxxxx message number. **Command:** The message text is one segment long and begins with the delimiter '/', followed by the command. **Command response:** The command response is usually a DFSxxxx message or one segment of a multisegment command.
CR	1	Carriage return character. This field is optional. If it is there, one byte is included in the LL. Input commands do not include a carriage control character.
Work area	20	Additional area AOIE can use to add text to the message.

Content of registers on exit

There is no requirement for exit registers. AOIE communicates using the reply field in the function-specific parameter list. AOIE is passed this list when it is entered. On exit, the registers contain the following:

Register	Contents
14	Return Address
15	0

Available work areas for the AOIE user exit

The AOIE user exit passes pointers to three different work areas:

Table 180. Work areas for the AOIE user exit

Work area	Description	Note
SXPLAWRK	The SXPLAWRK field in the standard user exit parameter list points to a 512-byte work area. This work area is shared among all AOIE user exits within one IMS system, which means the same work area address is passed to every AOIE exit call.	Be careful when using this work area because its content can be changed by other instances of calls to AOIE.

Table 180. Work areas for the AOIE user exit (continued)		
Work area	**Description**	**Note**
SXPLASWA	The SXPLASWA field in the standard user exit parameter list points to a 256-byte work area. This work area is unique to each AOIE exit module. If you have two AOIE exits defined, each module receives its own 256-byte work area address in SXPLASWA. The same address is passed to every call of the same module.	If your exit makes IMS user exit callable services calls or invokes any IMS system services directly, this work area is not recommended for use for dynamic (working) storage.
AOE0WRKA	The AOE0WRKA field in the function specific parameter list points to a 256-byte work area. This work area is unique to the IMS ITASK under which the AOIE exit is called, but is not unique among multiple user exit modules. All AOIE user exit modules get the same work area address when these modules are invoked serially under a given IMS ITASK. The same work area is passed for the segments of a message or for a command and the related command responses.	

Related concepts

IMS Automated Operator Interface (AOI) (Operations and Automation)

Related reference

"Type-1 Automated Operator exit routine (DFSAOUE0)" on page 438

The Type-1 Automated Operator exit routine (DFSAOUE0) is called continuously for operator-entered commands from terminals, APPC and OTMA devices, and the command responses to operator-entered commands. The AO exit routine intercepts these messages before IMS sends the system message, runs the command, or sends the command response.

"Routine binding restrictions" on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"IMS callable services" on page 13

IMS provides IMS callable services for exit routines to provide the user of the exit routine with clearly defined interfaces.

"IMS standard user exit parameter list" on page 5

Many of the IMS user exit routines are called with a standard interface, which allows the exit routines to access IMS control blocks with callable services.

Types of messages passed to this routine

IMS passes specific types of messages and commands to the Type 2 Automated Operator user exit routine (DFSAOE00 or another AOIE type user exit).

The following types of messages can be passed to the AOIE type user exit:

- IMS system messages destined for the master terminal (the z/OS system console in the DBCTL environment).
- IMS system messages being logged to the secondary master terminal.
- Commands entered from a terminal, with the following exceptions:
  - `/FORMAT`
  - `/LOOPTEST`
  - `/MSVERIFY`
  - `/RELEASE`
  - `/NRESTART` (sent to the exit routine, however, in the DBCTL environment)
  - `/ERESTART` (sent to the exit routine, however, in the DBCTL environment)

  IMS passes the command after basic edit and optional editing routines have had a chance to modify it. This modified input can contain carriage control characters.
- IMS command responses to the terminal.
- Commands issued from an AO application using the ICMD call.
- Commands generated internally by IMS.

IMS does not pass all system messages, operator-entered commands, and command responses to AOIE type user exit. The following are types of messages IMS does not pass to AOIE type user exit:

- Command responses to an AO application
- Commands issued from an AO application using the CMD call (including the responses to those commands)
- System messages for which the destination is not the master terminal (secondary master messages are passed if indicated during INIT call)
- Command responses to IMS internally generated commands

Subsections:

- "Changes the command editor makes" on page 472
- "Commands with network-qualified LU names" on page 473

Changes the command editor makes

The command editor translates certain control characters in any command you enter from a terminal or in any ICMD call. You need to accommodate this translation when writing your exit routine.

The translation is shown in the following table.

Table 181. Translation of control characters in commands

From	To
X'14' Restore	X'5D' Right parenthesis
X'15' New line	X'40' Blank
X'24' Bypass	X'4D' Left parenthesis
X'40' Blank	X'40' Blank
X'4B' Period	X'4B' Period
X'4D' Left parenthesis	X'4D' Left parenthesis
X'5D' Right parenthesis	X'5D' Right parenthesis
X'60' Dash	X'60' Dash

Table 181. Translation of control characters in commands (continued)

From	To
X'6B' Comma	X'6B' Comma
X'6D' Dash	X'40' Blank
X'7E' Equal	X'40' Blank

Commands with network-qualified LU names

If you use network-qualified LU names at your installation, the LU name for LU 6.2 application programs can be 17 bytes long. For IMS commands, the network-qualified LU names must be enclosed in single quotation marks (for example, 'NETID.LUNAME').

If an IMS command with the network-qualified LU name is passed to an AOIE type exit routine, IMS modifies the network-qualified LU name in the input command before the command is passed to the AOIE type exit routine. The single quotes around the network-qualified LU name are replaced with blanks, and the period separating the network-identifier and the LU name is replaced with a colon.

Example: A /DISPLAY command with a network-qualified LU name entered at the terminal as:

```
/DISPLAY LUNAME 'NETWORK1.LUNAME1' LUNAME2 INPUT
```

is passed to the AO exit routine or logged to the secondary master as:

```
/DISPLAY LUNAME  NETWORK1:LUNAME1  LUNAME2 INPUT
```

Processing when a system message is generated

The following figure shows processing when a system message is generated.

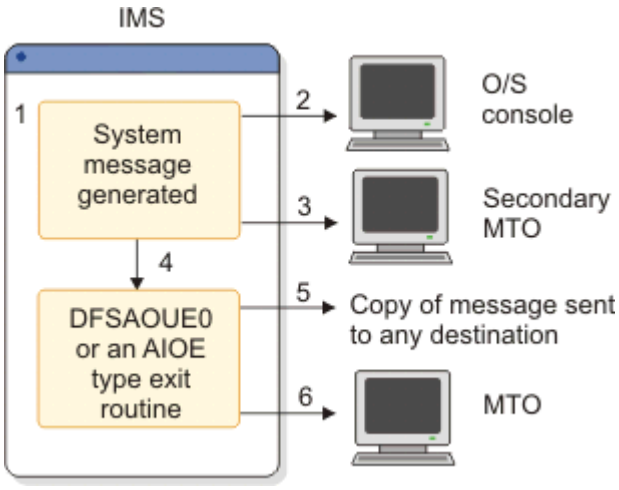


Figure 27. Processing when a system message is generated

Notes:

1. IMS generates a system message destined for the master terminal.
2. A copy of the message can be sent to the z/OS system console. This depends on the specific message and is determined by IMS.
3. A copy of the message can be sent to the secondary master terminal if it exists and if you have specified that this is to be done.
4. The copy of the message destined for the master terminal is passed to the AOIE type exit routine.

5. The AOIE type exit routine can send a copy of the message to an AO application. This is done by enqueuing the message to an AOI token. The AOIE type exit routine can alter or delete any segment of the message.
6. The message is sent (unless it has been deleted) to the master terminal.

   If both an AOIE type exit routine and DFSAOUE0 had been loaded, this picture would be conceptually the same. However, when the AOIE type exit routine got control it could either process the message, or it could return a code indicating DFSAOUE0 should be called to do the processing instead.

Commands entered at the terminal

The following figure shows processing when a command is entered at the terminal.

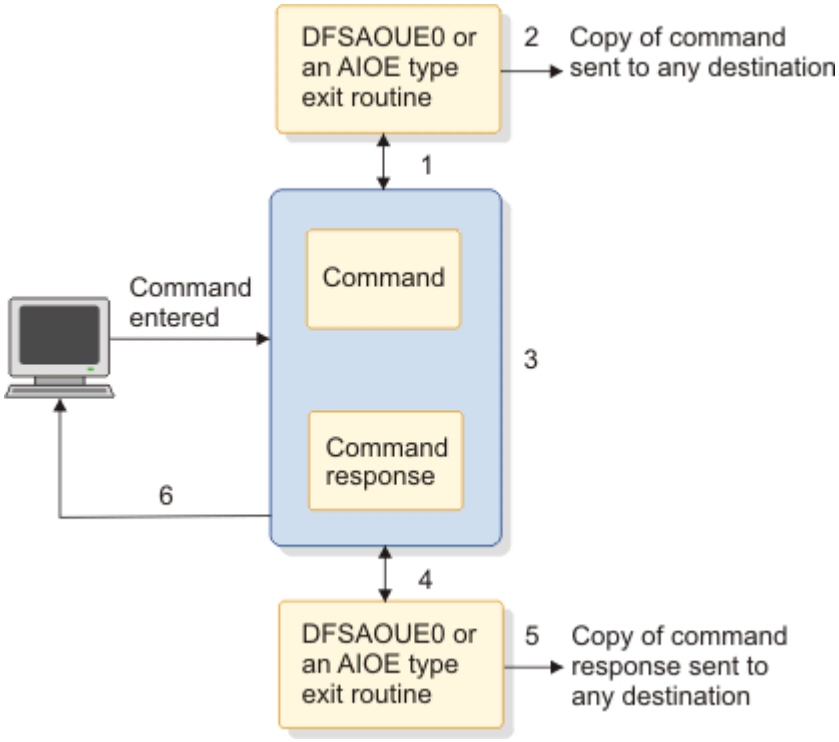


Figure 28. Processing when a command is entered at the terminal

Notes:

1. When a command is entered from a terminal, IMS sends a copy of the command to the AOIE type exit routine before executing the command.
2. The AOIE type exit routine can send a copy of the command to any AO application (using the AOI token).
3. IMS executes the command and generates a command response.
4. IMS passes the command response to the AOIE type exit routine. The AOIE type exit routine can send a copy of the command response to any AO application (using the AOI token).
5. The command response is sent to the terminal that originated the command.

   If multiple AO exit routines (one or more AOIE type exit routines and DFSAOUE0) had been loaded, this picture would be conceptually the same. However, when an AOIE type exit routine got control, it could either process the command or return a code indicating DFSAOUE0 should be called to do the processing instead.

Commands entered from an AO application

The following figure shows processing when a command is entered from an AO application.

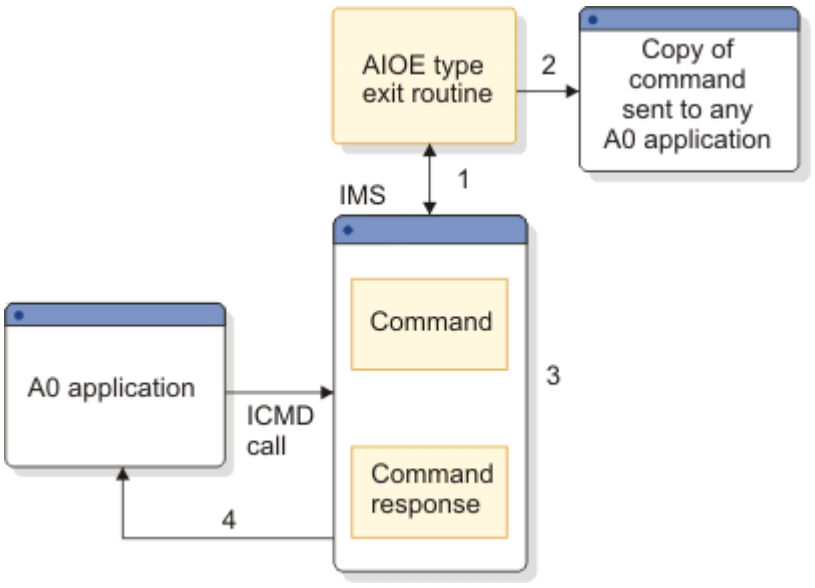


Figure 29. Processing when a command is entered from an AO application

Notes:

1. When a command is entered from an AO application using an ICMD call, IMS sends a copy of the command to the AOIE type exit routine before executing the command.
2. The AOIE type exit routine can send a copy of the command to any AO application (using the AOI token).
3. IMS executes the command and generates a command response.
4. IMS sends the command response back to the AO application.

   The type 1 AO exit routine (DFSAOUE0) cannot process commands entered from an AO application.

User Message table (DFSCMTU0)

You can create your own messages and list them in the User Message table (DFSCMTU0).

Although there are IMS system message tables containing messages that IMS returns to edit and exit routines, these messages might not be appropriate for your installation's needs. If this is the case, you can create your own messages and list them in your own message table.

- "About this table" on page 475

About this table

IMS assigns the prefix of the user text from the message table with DFSUxxx, where *xxx* is the message number. You can then use this message table with the following user edit and exit routines:

- Command Authorization exit routine (DFSCCMD0)
- Front-End Switch exit routine (DFSFEBJ0)
- Global Physical Terminal edit routine (DFSGPIX0)
- Logoff exit routine (DFSLGFX0)
- Message Switching Input edit routine (DFSCNTE0)
- Input Message Segment edit routine (DFSME127)
- Physical Terminal Input edit routine (DFSPIXT0)
- Queue Space Notification exit routine (DFSQSPC0)
- Signon/off Security exit routine (DFSCSGN0)

- Signon exit routine (DFSSGNX0)
- Signoff exit routine (DFSSGFX0)
- Transaction Authorization exit routine (DFSCTRN0)
- Transaction Code Input edit routine (DFSCSMB0)

The following table shows the attributes of the User Message table.

Table 182. User message table attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name the message table module **DFSCMTU0**, assemble it, and specify it in startup procedures. It must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST concatenation.
Link editing	No special steps are required to include this table.
Including the routine	You need to specify USERMSGS= in startup procedures. The user message table module must be placed in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.SDFSRESL.
IMS callable services	IMS callable services are not applicable for use with this table.
Sample routine location	No sample available.

Restriction: This table cannot be used in a DBCTL environment.

Writing a routine to use this table

In order for a routine to use the messages you've placed in the message table, you'll need to choose a key that represents a message number in the table. In the case of the Queue Space Notification exit routine (DFSQSPC0) and the Signon exit routine (DFSSGNX0), the negative value of the message key needs to be placed in register 15 on return from the routine. For the other exit routines listed in the previous topic, the positive value of the message key needs to be placed in register 1 on return from the routine along with a specific return code in register 15.

Exception: The Front-End Switch exit routine (DFSFEBJ0) is an exception to this. Refer to the descriptions of each exit routine for more information.

Formatting the table

The format of your message table (DFSCMTU0) must be as follows:

- The table must start with the instruction BALR 15,14.
- Message numbers range from 1 to (and including) 999, in ascending sequence.
- The maximum size for the text of each message is 128 characters, If the message text exceeds 78 characters, it will be truncated if it is sent to a 3270 terminal that is formatted using IMS-supplied default formats.
- The message length must be an even value (otherwise, an erroneous character might appear following the last character of the text).
- Exclude device control characters from the text of your messages. IMS always adds NEW LINE (NL) control characters to the beginning and end of each message. For messages processed by the Message Format Service (MFS), the device control characters will be changed to X'00' for a 3270 display and to X'40' for other devices.
- Each message entry must start on a half-word boundary. The entry format is:

```
label   DC  H'message number'
        DC  AL2 (entry length including number
```

```
             and length fields)
         DC  C'message text of even length'
```

- An entry with message number X'7FFF' signals the end of the message table.

Sample user message table and routine

You can review the sample user message table and routine to understand how to change the text of the messages issued by a revised version of the Queue Space Notification exit routine.

This example shows how you would use user message tables to change the text of the messages issued by a revised version of the Queue Space Notification exit routine.

Subsections:

- "Sample table" on page 477
- "Sample routine" on page 477

Sample table

The following table sample contains the messages specified by an IMS user and has been included in the IMS system.

```
DFSCMTU0 CSECT
*                               USER MESSAGE TABLE FOR USER QUEUE
*                               SPACE NOTIFICATION EXIT EXAMPLE.

         BALR  15,14
M013     DC    H'513'            QMGR0
         DC    AL2(M014-M013)
         DC    C'RECORDS IN QBLKS DATASET EXCEED UPPER THRESHOLD '
M014     DC    H'514'            QMGR0
         DC    AL2(M015-M014)
         DC    C'RECORDS IN SMSGQ DATASET EXCEED UPPER THRESHOLD '
M015     DC    H'515'            QMGR0
         DC    AL2(M016-M015)
         DC    C'RECORDS IN LMSGQ DATASET EXCEED UPPER THRESHOLD '
M016     DC    H'516'            QMGR0
         DC    AL2(M017-M016)
         DC    C'RECORDS IN QBLKS DATASET BELOW LOWER THRESHOLD'
M017     DC    H'517'            QMGR0
         DC    AL2(M018-M017)
         DC    C'RECORDS IN SMSGQ DATASET BELOW LOWER THRESHOLD'
M018     DC    H'518'            QMGR0
         DC    AL2(M999-M018)
         DC    C'RECORDS IN LMSGQ DATASET BELOW LOWER THRESHOLD'
M999     DC    X'7FFF'
         END   ,
```

Sample routine

In this sample routine, the IMS-supplied exit routine (DFSQSPC0) has been replaced by a modified version of the routine that a user has written. The user-modified DFSQSPC0 has the following characteristics:

1. Existing IMS message equates have been replaced by user equates.
2. The list of messages used by the routine code has been changed to refer to the user messages.
3. Load Negative Register (LNR) instructions have been added to store the negative of the user message key in register 15 before returning to the caller of DFSQSPC0. This causes IMS to look in the User Message Table (DFSCMTU0) rather than the system tables for the text of the message.

The following sample shows the modified Queue Space Notification exit routine, DFSQSPC0:

```
*********************************************************************
*                                                                   *
*             M O D U L E     P R O L O G                           *
*                                                                   *
*********************************************************************
*                                                                   *
```

```
* MODULE NAME: DFSQSPC0                                                 *
*                                                                       *
* DESCRIPTIVE NAME: SAMPLE USER QUEUE SPACE NOTIFICATION EXIT           *
*                                                                       *
* FUNCTION:                                                             *
*                                                                       *
*    INTERROGATES NUMBER OF RECORDS CURRENTLY IN USE FOR A              *
*    DATASET AND DETERMINES WHETHER OR NOT TO DISPLAY                   *
*    THRESHOLD MESSAGES (SEE OUTPUT)                                    *
*                                                                       *
* NOTES:                                                                *
*                                                                       *
*    RESTRICTIONS:                                                      *
*                                                                       *
*    DFSQSPC0 MUST NOT IWAIT. THERE IS ONLY ONE PARAMETER AREA          *
*    (IN QPOOL), HENCE THE QUEUE MANAGER MUST NOT IWAIT BETWEEN         *
*    THE TIME IT SETS UP THE PARAMETER LIST AND THE TIME IT NO          *
*    LONGER NEEDS IT, FOLLOWING INVOCATION OF THE EXIT (DFSQSPC0)       *
*                                                                       *
*    IN ORDER TO UPDATE THE "IN USE" COUNT WITHOUT FIRST ZEROING        *
*    THE HIGH ORDER BYTE THE HIGH ORDER BIT OF THE FLAG BYTE MUST       *
*    ALWAYS BE 0.                                                       *
*                                                                       *
*    DEPENDENCIES: NONE                                                 *
*                                                                       *
*    REGISTER CONVENTIONS: STANDARD IMS                                 *
*                                                                       *
*    MODULE TYPE:                                                       *
*                                                                       *
*    IMS DC - QUEUE MANAGER EXIT (MAY BE REPLACED BY USER EXIT)         *
*                                                                       *
*    ATTRIBUTES: REENTRANT                                              *
*                                                                       *
* ENTRY POINT: DFSQSPC0                                                 *
*                                                                       *
*    PURPOSE: SEE FUNCTION                                              *
*                                                                       *
*    LINKAGE: BALR R14,R15 FROM DFSQMGR0 WHENEVER AN LRECL (DRRN)       *
*    IS ASSIGNED OR FREED                                               *
*    NOTE: IN ORDER TO REDUCE THE NUMBER OF INSTRUCTIONS IN THE         *
*    EXIT ONLY THE WORK REGISTERS (4 AND 5) ARE SAVED AND RESTORED      *
*                                                                       *
* INPUT:                                                                *
*                                                                       *
*    R0  =  DATASET INDICATOR                                           *
*           00 IF QBLKS                                                 *
*           04 IF SMSGQ                                                 *
*           08 IF LMSGQ                                                 *
*                                                                       *
*    R2  =  POINTER TO PARAMETER LIST...                                *
*           1ST WORD, 1ST BYTE                                          *
*                    = FLAG CONTAINING BIT (X'40') THAT INDICATES       *
*                      WHETHER (ON) OR NOT (OFF) # OF RECORDS IN        *
*                      USE EXCEEDED THE UPPER THRESHOLD BUT HAS         *
*                      NOT YET DROPPED BELOW THE LOWER THRESHOLD        *
*           1ST WORD, 2ND-4TH BYTE                                      *
*                    = # OF RECORDS CURRENTLY IN USE                    *
*           2ND WORD = MAX # OF RECS ASSIGNABLE BEFORE SHUTDOWN         *
*                                                                       *
*    R10 = SCD ADDRESS                                                  *
*                                                                       *
*    R14 = RETURN ADDRESS                                               *
*                                                                       *
*    THE UPPER AND LOWER THRESHOLD VALUES ARE OBTAINED FROM SCD         *
*    FIELDS SCDQTU AND SCDQTL. THE QUEUE MANAGER INITIALIZATION         *
*    MTHE JCL USED TO BRING UP THE CONTROL REGION.                      *
*    CORRESPONDING FIELDS IN RGPARMS. RGQTU AND RGQTL ARE               *
*    1) IMS DEFAULTS (75% AND 60%), 2) USER DEFAULTS ESTABLISHED        *
*    BY SPECIFYING VALUES FOR QTU AND QTL IN THE DFSPBxxx MEMBER        *
*    OR 3) EXEC PARAMETER VALUES FOR QTU AND QTL ON                     *
*    THE JCL USED TO BRING UP THE CONTROL REGION.                       *
*                                                                       *
* OUTPUT:                                                               *
*                                                                       *
*    R0, R2, AND R10 ARE UNCHANGED                                      *
*    R15 = RETURN CODE                                                  *
*        = 0 IF NO THRESHOLDS PASSED                                    *
*        = ONE OF THE FOLLOWING MESSAGE KEYS IF THRESHOLD PASSED:       *
*                                                                       *
DFS513   EQU  513   RECS IN QBLKS EXCEED UPPER THRESHOLD                *
*          IF QBLKS UPPER THRESHOLD EXCEEDED (BIT X'40' IN              *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED ON)                  *
```

```
*                                                                     *
DFS514   EQU  514   RECS IN SMSGQ EXCEED UPPER THRESHOLD              *
*          IF SMSGQ UPPER THRESHOLD EXCEEDED (BIT X'40' IN           *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED ON)               *
*                                                                    *
DFS515   EQU  515   RECS IN LMSGQ EXCEED UPPER THRESHOLD             *
*          IF LMSGQ UPPER THRESHOLD EXCEEDED (BIT X'40' IN           *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED ON)               *
*                                                                    *
DFS516   EQU  516   RECS IN QBLKS DATASET BELOW LWR THRESHOLD        *
*          IF QBLKS LOWER THRESHOLD PASSED (BIT X'40' IN             *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED OFF)              *
*                                                                    *
DFS517   EQU  517   RECS IN SMSGQ DATASET BELOW LWR THRESHOLD        *
*          IF SMSGQ LOWER THRESHOLD PASSED (BIT X'40' IN FLAG        *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED OFF)              *
*                                                                    *
DFS518   EQU  518   RECS IN LMSGQ DATASET BELOW LWR THRESHOLD        *
*          IF LMSGQ LOWER THRESHOLD PASSED (BIT X'40' IN             *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED OFF)              *
*                                                                    *
*    NOTE: IF DFSQSPC0 IS REPLACED BY A USER EXIT THE USER           *
*          MESSAGE NUMBERS MUST BE RETURNED IN R15 AS THE            *
*          NEGATIVE OF THE POSITIVE MESSAGE NUMBER (LNR).            *
* NORMAL EXIT: SEE OUTPUT                                            *
*                                                                    *
* ERROR EXIT: NONE                                                   *
*                                                                    *
* EXTERNAL REFERENCES: NONE                                          *
*                                                                    *
* CHANGE ACTIVITY: SEE CHANGEID                                      *
*                                                                    *
**********************************************************************
         EJECT
**********************************************************************
*
*    PSEUDO CODE
*
**********************************************************************
*                                                                    *
*                                                                    *
*    IF THE BIT IN THE PARAMETER FLAG INDICATING # OF RECORDS        *
*         EXCEEDED UPPER THRESHOLD IS ON                             *
*                                                                    *
*         THEN                                                       *
*         IF # OF RECORDS CURRENTLY IN USE HAS DROPPED BELOW LOWER   *
*         THRESHOLD (60% OF MAXIMUM # OF ASSIGNABLE RECORDS BEFORE   *
*         SHUTDOWN)                                                  *
*                                                                    *
*              THEN                                                  *
*              TURN OFF BIT IN PARAMETER FLAG INDICATING # OF RECORDS *
*              EXCEEDED UPPER THRESHOLD.                             *
*              SET R15 = KEY FOR MESSAGE INDICATING DATASET OK AGAIN. *
*                                                                    *
*              ELSE                                                  *
*              NULL                                                  *
*                                                                    *
*         ELSE                                                       *
*         IF # OF RECORDS CURRENTLY IN USE > UPPER THRESHOLD (75% OF  *
*              MAXIMUM # OF ASSIGNABLE BYTES BEFORE SHUTDOWN)        *
*                                                                    *
*              THEN                                                  *
*              SET BIT IN PARAMETER FLAG INDICATING # OF RECORDS     *
*              EXCEEDED UPPER THRESHOLD.                             *
*              SET R15 = KEY FOR MESSAGE INDICATING UPPER THRESHOLD  *
*              EXCEEDED.                                             *
*                                                                    *
*              ELSE                                                  *
*              NULL                                                  *
*                                                                    *
*    RETURN TO CALLER.                                               *
*                                                                    *
**********************************************************************
         EJECT
DFSQSPC0 CSECT
         CHANGEID NAME=DFSQSPC0&SYSDATE,BASE=R12,LINKAGE=IMS,         X
               SAVE=(4,,5,,12,)
         CHANGEID IDEND=YES
*
         USING PARM,R2       ADDRESSABILITY TO PARM AREA.
         USING SCD,R10       ADDRESSABILITY TO SCD.
*
```

```
         TM    PFLAG,PFEXCD  EXCEEDED UPPER THRESHOLD?
         BZ    QSPC100       NO... CONTINUE.
*
         L     R5,PMAX       CALCULATE LOWER THRESHOLD
         M     R4,SCDQTL     (60% OF MAXIMUM NUMBER
         D     R4,=F'100'    OF RECORDS ASSIGNABLE).
*
         L     R4,PINUSE     GET FLAG + IN USE COUNT.
         LA    R4,0(,R4)     GET RID OF FLAG.
         CR    R5,R4         LOWER THRESHOLD : # CRNTLY IN USE
         BNH   RETURN        BR IF LOWER THRESHLD <= CUR IN USE.
*
         NI    PFLAG,X'FF'-PFEXCD TURN OFF EXCEEDED FLAG.
         LR    R15,R0        SET UP
         SRL   R15,1         INDEX.
         LH    R15,MSGTBL1(R15)   GET APPROPRIATE MESSAGE KEY.
         LNR   R15,R15       INDICATE USER MESSAGE KEY.
         B     RETURN1
*
         L     R5,PMAX       CALCULATE UPPER THRESHOLD
         M     R4,SCDQTU     (75% OF MAXIMUM NUMBER
         D     R4,=F'100'    OF RECORDS ASSIGNABLE).
*
         L     R4,PINUSE     GET FLAG + IN USE COUNT.
         LA    R4,0(,R4)     GET RID OF FLAG.
         CR    R5,R4         UPPER THRESHOLD : # CURNTLY IN USE
         BNL   RETURN        BR IF UPPER THRESHLD >= CUR IN USE.
*
         OI    PFLAG,PFEXCD  SHOW CURRENT IN USE EXCEEDED MAX.
         LR    R15,R0        SET UP
         SRL   R15,1         INDEX.
         LH    R15,MSGTBL2(R15)   GET APPROPRIATE MESSAGE KEY.
         LNR   R15,R15       INDICATE USER MESSAGE KEY.
         B     RETURN1
*
RETURN   EQU   *
         SR    R15,R15       RC FOR NO THRESHOLDS PASSED.
*
RETURN1  EQU   *
         LEAVE RC=(15),RESTORE=(4,,5,,12,)
         EJECT
*********************************************************************
*        MESSAGES RETURNED WHEN THRESHOLDS PASSED
*
*********************************************************************
*
MSGTBL1  DS    0H
         DC    AL2(DFS516)   RECS IN QBLKS BELOW LOWER THRESH
         DC    AL2(DFS517)   RECS IN SMSGQ BELOW LOWER THRESH
         DC    AL2(DFS518)   RECS IN LMSGQ BELOW LOWER THRESH
MSGTBL2  DS    0H
         DC    AL2(DFS513)   RECS IN QBLKS EXCEED UPPER THRESH
         DC    AL2(DFS514)   RECS IN SMSGQ EXCEED UPPER THRESH
         DC    AL2(DFS515)   RECS IN LMSGQ EXCEED UPPER THRESH
         EJECT
         LTORG
         EJECT
         REQUATE
         PRINT NOGEN
         ISCD
         PRINT GEN
         EJECT
         DFSPARM
         END
```

XRF Hardware Reserve Notification exit routine

Use the XRF Hardware Reserve Notification exit routine to receive notification of z/OS reserves that should not be converted to global enqueues.

This topic contains Product-sensitive Programming Interface information.

Subsections:

About this routine

This exit routine must be written as reentrant. No default is provided.

The exit name is set by IMS to DFS.XRFRESERVE. The XRF Hardware Reserve Notification Exit uses the Dynamic Exit Facility. The Dynamic Exit Facility uses DFS.XRFRESERVE for its EXITNAME parameter. You can provide any name for the exit routine itself.

The following table shows the attributes of the XRF Hardware Reserve Notification exit routine.

Table 183. XRF Hardware Reserve Notification exit routine attributes

Attribute	Description
IMS environments	XRF alternate
Naming convention	The exit name is set to DFS.XRFRESERVE. You can provide any name for the exit routine. Define the name of the exit routine to the z/OS dynamic exit facility by using either the PROGxx parmlib member EXIT statement or the SETPROG EXIT operator command. See the "Dynamic Exits Facility" topic in *z/OS MVS Installation Exits* manual.
Binding	You can bind the exit routine into: • A data set that becomes part of the PLPA, MLPA, or FLPA during initial program load. • A data set that is part of the LNKLST concatenation. • The nucleus initialization program IEANUC0x • Any PDS or PDSE that is designated by the DSNAME option of: – The SETPROG EXIT command – The EXIT ADD statement of a PROGxx parmlib member
Including the routine	No additional steps are needed to include this routine.
IMS callable services	This exit routine cannot use IMS Callable Storage Services.
Sample routine location	No sample exit routine is provided.

Attributes of the routine

The XRF Hardware Reserve Notification exit routine must be written as reentrant. This exit routine receives control running in 31-bit addressing mode and it must return control in that mode. The exit routine is called in TASK mode, key 7, with no locks held, and in non-cross memory, non-AR mode.

Calling the routine

The exit routine is called whenever IMS reserves or releases one or more volumes that contain log data during XRF takeover. The reserve call is made prior to the actual reserve and the release call is made after the actual release.

Communicating with IMS

IMS communicates with this routine through the entry registers and a parameter list.

Content of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following content:

Register	Content
1	Address of the standard exit parameter list.

Register	Content
13	Address of the standard save area.
14	Return address to Dynamic Exit Service.

Exit parameter list

The parameter list contains the addresses of four parameters:

Parameter	Definition
1	A fullword containing the version number of the parameter list.
2	A fullword containing a function code. Two functions are defined: • FRBFNRSV EQU 1 ... function = reserve • FRBFNREL EQU 2 ... function = release
3	A list of devices being reserved or released. The list format is a halfword containing the number of devices in the list followed by that number of halfword device addresses. These addresses are obtained from the UCBCHAN field of the UCBs involved.
4	An eight character field containing the RSENAME.

Content of registers on exit

The Dynamic Exit Facility will restore IMS register content.

Related concepts

z/OS: Dynamic Exits Facility

Related reference

“Routine binding restrictions” on page 9

If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Part 2. Base Primitive Environment-based exit routines

Use these topics to design and write user-supplied modules for exit routines that are supported by Base Primitive Environment (BPE) interfaces and services.

Chapter 5. BPE user-supplied exit routine interfaces and services

BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

This topic describes the Base Primitive Environment (BPE) user exit routine interfaces and services in detail.

This topic contains Product-sensitive Programming Interface information.

Note: Throughout this topic the term "user exit routine" means "user-supplied exit routine."

General BPE user-supplied exit routine interface information

Some IMS components (for example, CQS, OM, RM, and SCI) use BPE services to define and manage calls to user exit routines. BPE also has its own user exit routines. BPE also provides a common user exit routine run time environment. The run time environment includes the following:

- A standard BPE user exit parameter list
- Static work areas for the routines
- Dynamic work areas for the routines
- Callable services for the routines
- A recovery environment to protect against abends in the user exit routines

Recommendation: Write BPE user exit routines in assembler, not in a high level language. BPE does not support exit routines that run under Language Environment for z/OS. If you write an exit routine in a high level language, and that routine runs in the Language Environment for z/OS, you might have abends or performance problems. Language Environment for z/OS is designed for applications that run in key 8, problem program state. BPE user exit routines run in key 7 supervisor state.

Standard BPE user exit parameter list

All BPE-managed user exit routines receive a pointer to a Standard BPE user exit parameter list in R1. The format of this parameter list is the same for all exit routines, and is mapped by the BPEUXPL DSECT (in the BPEUXPL macro). The following table provides the following information about the fields in the Standard BPE user exit parameter list:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Table 184. Standard BPE user exit parameter list

Field name	Offset	Length	Field usage	Description
BPEUXPL	X'00'	N/A	N/A	DSECT label for Standard BPE user exit parameter list.

Table 184. Standard BPE user exit parameter list (continued)

Field name	Offset	Length	Field usage	Description
UXPL_VERSIONP	X'00'	X'04'	Input	Pointer to a word containing the Standard BPE user exit parameter list version number. The current version of the parameter list is X'00000002'. (EQU symbol UXPL_VER2.)
UXPL_CSTOKENP	X'04'	X'04'	Input	Pointer to the BPE callable services token.
UXPL_STATICWAP	X'08'	X'04'	Input	Pointer to a 256-byte static work area. Each exit routine module is assigned its own static work area. The contents of the static work area are preserved from call to call.
UXPL_DYNAMICWAP	X'0C'	X'04'	Input	Pointer to a 512-byte dynamic work area. This area is intended as working storage for a user exit routine for the duration of that exit routine's run. The contents of this area are not preserved from call to call.
UXPL_EXITPLP	X'10'	X'04'	Input	Pointer to an exit-type-specific parameter list. The exit-type-specific parameter list contains fields that are unique to the type of exit routine being called.
UXPL_CALLNEXTP	X'14'	X'04'	Input	Pointer to a byte of storage that the user exit routine can use to indicate whether to call other subsequent exit routines of the same type for the current instance of the exit routine call.

Table 184. Standard BPE user exit parameter list (continued)

Field name	Offset	Length	Field usage	Description
UXPL_COMPTYPEP	X'18'	X'04'	Input	Pointer to a 4-byte character string containing the IMS component type for the address space in which the exit routine is being called. The string is left-justified and padded with blanks as needed to make it a 4-byte string. Possible values are: **CQS** Common Queue Server **DBRC** Database Recovery Control **HWS** IMS Connect **OM** Operations Manager **REPO** IMS Repository Server **RM** Resource Manager **SCI** Structured Call Interface **ODBM** Open Database Manager **Important:** This field is present only when the word pointed to by UXPL_VERSIONP is equal to the value of UXPL_VER2 or greater.
UXPL_COMPVERP	X'1C'	X'04'	Input	Pointer to a 3-byte field in storage containing the internal version number of the IMS component for the address space in which the exit routine is being called. The version number is of the form *vvrrpp*, where: ***vv*** Component version number. ***rr*** Component release number. ***pp*** Component point release number. **Important:** This field is present only when the word pointed to by UXPL_VERSIONP is equal to the value of UXPL_VER2 or greater.

Table 184. Standard BPE user exit parameter list (continued)

Field name	Offset	Length	Field usage	Description
UXPL_BPEVERP	X'20'	X'04'	Input	Pointer to a 3-byte field in storage containing the BPE internal version number for the address space in which the exit routine is being called. The version number is of the form *vvrrpp*, where: ***vv*** BPE version number. ***rr*** BPE release number. ***pp*** BPE point release number. **Important:** This field is present only when the word pointed to by UXPL_VERSIONP is equal to the value of UXPL_VER2 or greater.
UXPL_SYSIDP	X'24'	X'04'	Input	Pointer to an 8-character system ID. The system ID is a character ID string that may be used by the IMS component using BPE services (for example, the CQS ID that is derived from the CQS SSN= startup parameter). If the IMS component has not provided a system ID to BPE, then the field pointed to by this pointer will be all blanks. If the system ID is shorter than eight characters, it is padded on the right with blanks to make it eight characters. **Important:** This field is present only when the word pointed to by UXPL_VERSIONP is equal to the value of UXPL_VER2 or greater.

Work areas provided by BPE

Each user exit routine is passed two work areas by BPE every time the exit routine is called. The two work areas are:

- The static work area
- The dynamic work area

Static work area

The static work area is pointed to by field UXPL_STATICWAP in the Standard BPE user exit parameter list. The static work area is 256 bytes in length. Each user exit routine is assigned its own static work area that is not shared between exit routines of the same type. The same work area is passed every time a particular user exit routine is called, and the contents of the work area are preserved from call to call. A user exit routine can use the static work area to save data between calls to the exit routine. The static work area is cleared (set to zeros) the first time a user exit routine is invoked.

When a user exit routine is refreshed with the REFRESH USEREXIT command, the same static work area continues to be passed to the new copy of the module that was being passed to the old copy. If a user exit routine is removed from an EXITDEF list and a REFRESH USEREXIT command is issued, the static work

area for the module is deleted. If the exit module is then later added back to the EXITDEF list and another REFRESH USEREXIT command is issued, the exit routine gets a new (cleared) static work area.

Dynamic work area

The dynamic work area is pointed to by field UXPL_DYNAMICWAP in the Standard BPE user exit parameter list. The dynamic work area is 512 bytes in length. The dynamic work area is used as working storage by a user exit routine for the current call only. The dynamic storage area's address might not be the same, nor are its contents preserved from call to call. The dynamic work area is not cleared when a user exit routine receives control; therefore, the work area might contain residual data.

Related reference

BPE USEREXIT commands (Commands)

BPE exit list members of the IMS PROCLIB data set (System Definition)

"BPE Initialization-Termination user-supplied exit routine" on page 509
The BPE Initialization-Termination user-supplied exit routine is called during BPE initialization and normal BPE termination.

"BPE Statistics user-supplied exit routine" on page 511
The BPE Statistics user-supplied exit routine is called at regular intervals during the life of a BPE address space, and a final time at normal address shutdown, to gather address-space related statistics.

Calling subsequent exit routines in BPE

Each user exit routine type can have multiple exit routine modules associated with it and can decide whether subsequent exit routines in the list that are to be called on return to BPE.

Each user exit routine type can have multiple exit routine modules associated with it. By default, BPE calls each module in the order that it was specified on the EXITS parameter of the EXITDEF= statement. However, some exit types call the specified modules in reverse order. If an exit type calls modules in reverse order, that will be explicitly stated in the exit's individual documentation. The EXITDEF= statement of the BPE user exit PROCLIB member defines the list of exit routines.

Each user exit routine can decide whether subsequent exit routines in the list that are to be called on return to BPE. For example, a list of exit routines are called to make a decision about processing for a particular resource. If exit routine ABC cannot make the decision, it can return an indication that the next exit routine in the list, routine DEF, is to be called so that it can try to make the decision. If exit routine ABC is able to make the decision, it can return an indication that the next exit routine in the list, routine DEF, need not be called because the decision has already been reached.

Field UXPL_CALLNEXTP in the Standard BPE user exit parameter list is a pointer to a byte in storage that the user exit routine can use to indicate whether to call the next exit routine in the list. If the exit routine does not set this byte, the default is to call the next exit routine in the list. If the exit routine sets this byte, it must set it to one of the following values, defined by EQUs in the BPEUXPL macro:

UXPL_CALLNEXTYES
Call the next exit routine in the list.

UXPL_CALLNEXTNO
Do not call the next exit routine in the list.

BPE may ignore the value of the UXPL_CALLNEXTP byte for certain types of user exit routines. In this case, all modules in the EXITDEF list for that exit type are always called. Exit types that ignore the UXPL_CALLNEXTP setting will explicitly state this information in their individual exit descriptions. If no information is given, the default condition is that the exit type will use the UXPL_CALLNEXTP setting.

Attention: Only UXPL_CALLNEXTYES and UXPL_CALLNEXTNO are defined values for this byte. Results are unpredictable if a user exit routine sets this byte to any value other than those listed here.

BPE user-supplied exit routine environment

BPE user-supplied exit routines are given control under certain environmental conditions.

All user exit routines are given control in the following environment unless otherwise stated:

Authorization
Supervisor state, PSW key 7

Dispatchable unit mode
TCB

Cross-memory mode
None (PASN=HASN=SASN)

AMODE
31

ASC mode
Primary

Interrupt Status
Enabled

Locks
None

All user exit routines receive control with the following registers set:

Register
Contents

R1
Pointer to Standard BPE user exit parameter list.

R13
Pointer to the first of two pre-chained save areas. The user exit routine can use the first save area to save the registers of its caller, and can use the second save area for lower-level calls that it makes. The save areas are chained together using standard z/OS save area linkage conventions.

R14
Return address.

R15
Entry point of exit routine.

Attention: Control must be returned to the return address passed to the user exit routine in R14. R15 can be set to a return code if appropriate for the specific exit routine type being called. Ensure that all other registers are restored to the values they had when the exit routine was called.

The contents of the registers not listed here are unknown and unpredictable.

Ensure that your user exit routines do not modify any fields in any parameter list that are not explicitly documented as output fields. The results of modifying non-output fields are unpredictable.

Write your user exit routines so that they are reentrant. User exit routines in the same EXITS= list are called serially within one occurrence of a call for that exit routine type. However, it is possible for a user exit routine to be entered simultaneously for different occurrences of a call, under different TCBs, for the same exit routine type.

An exit routine receives the same static work area, but receives another dynamic work area for each call when it is entered simultaneously. Be careful when updating fields in the static work area. They might be in the process of being changed by other instances of your exit routine module that are running in parallel.

BPE user exit routine performance considerations

Code user-supplied exit routines in ways that minimize path length and processing time as much as possible.

Some user exit routines might be called from mainline processing code. The amount and type of processing that is done by those exit routines can directly contribute to the total path length and time required to complete a unit of work.

Recommendations:

- Code your user exit routines in assembler language for the best performance. If you write exit routines in other languages, you might have performance problems. BPE does not support exit routines that run under Language Environment for z/OS.
- Use a BPE callable service when possible rather than the operating system equivalent, because the callable service is usually optimized to perform more efficiently in a BPE sub-dispatching environment.
- Operating system WAITs, SVCs, and I/O can all contribute to poor performance and should be used sparingly.

Abends in BPE user-supplied exit routines

BPE establishes a recovery environment before it calls user exit routines.

In most cases, BPE recovers from any abends that occur while a user exit routine is in control, and calls the next exit routine in the list, if any is indicated. When a user exit routine abends, BPE ignores any value that the abending exit routine may have set in the byte pointed to by UXPL_CALLNEXTP. BPE resets this byte to UXPL_CALLNEXTYES and then calls the next exit routine in the list.

BPE keeps a count of the number of abends that have occurred in each user exit routine module. The first time an abend occurs in a module, BPE issues a request to create an SDUMP to capture diagnostic information about the abend. BPE also creates a SYS1.LOGREC entry for the abend and issues the message, BPE0019E, indicating which exit routine module had control when the abend occurred. For subsequent abends in an exit routine module, BPE creates a SYS1.LOGREC entry and issues the message, BPE0019E, but does not issue the request to create an SDUMP.

When the number of abends indicated by the ABLIM parameter has been reached, BPE stops calling the abending exit routine module. The ABLIM parameter is specified as part of the EXITDEF= statement for that type of exit routine. The default value for the ABLIM parameter is 1 (to stop calling the exit routine after the first abend). You can change this value as required. The abend count for an exit routine is reset to zero if the exit routine type is refreshed.

Related reference

BPE exit list members of the IMS PROCLIB data set (System Definition)

BPE REFRESH USEREXIT command (Commands)

BPE user-supplied exit routine callable services

A set of callable services is provided that can be used by BPE-managed user exit routines to request certain functions from BPE. Callable services are requested by using the BPEUXCSV macro.

Recommendation: Choose the BPE service when there is a choice between using an operating system service or an equivalent BPE callable service. All callable services are Product-Sensitive Programming Interfaces (PSPIs).

Subsections:

- “BPEUXCSV macro description” on page 492
- “BPEUXCSV macro syntax” on page 493
- “Return from BPEUXCSV” on page 495

BPEUXCSV macro description

The purpose of the BPEUXCSV macro is to issue BPE callable service requests from a user exit routine called from a BPE environment. You can use this macro **only** for BPE-called exit routines (exit routines that are passed the address of a Standard BPE user exit parameter list in R1). BPE provides callable services that include the following functions:

- Get and free storage associated with the primary BPE TCB (usually job step). Some user exit routines can run under a different TCB each time they are called. Normally, storage obtained with GETMAIN is associated with the current TCB. If an exit routine obtained storage when it was called under one TCB and tried to free it when running under a different TCB, the storage free attempt may fail. The get storage and free storage callable services allow exit routines to get an area of storage when running under one TCB and to free it when running under a different TCB.
- Load and delete modules and associate these modules with the primary BPE TCB. Like the storage get and free services, the load and delete services handle module management when loaded and deleted from different TCBs.
- Get, retrieve, and free named storage areas. A named storage area is an area of storage that is associated with a 16-byte name. The address of the storage area can be retrieved given the name of the area. This allows different user exit routines to communicate with one another by using a common name for a shared named storage area.

When a callable service is invoked, the service may have to wait for the completion of some event. Depending on the environment at the time your user exit routine is called, such a wait can be either an OS WAIT (that is, the current TCB is suspended until the event completes) or a BPE-internal wait. For BPE-internal waits, BPE can run other ready work under the current TCB while your user exit routine is waiting for the event to complete. When the event does complete, BPE re-dispatches your exit routine's unit of work and completes the callable service request.

The possibility of waiting introduces the following situations, which your exit routine must be able to manage.

- Depending on the nature of the specific user exit routine (where and when it is called), your exit routine might be entered again for another exit routine call while the first instance of the exit routine is still waiting in a callable services request. Note that multiple concurrent calls to user exit routines are, in general, always possible. However, some user exit routines might normally be TCB-serialized (that is, their callers always run under a single TCB); these TCB-serialized routines might be entered multiple times when you use a callable service.
- Again, depending on the specific user exit routine, your exit routine might have control passed back from the BPE callable service request running under a different TCB than when it was originally called. This is because BPE provides the ability for a program that is using BPE services (such as CQS) to define a pool of TCBs. In this situation any TCB in the pool can run any unit of work that is assigned to the pool. So, your exit routine might be running under one TCB in a pool, make a callable services request, wait, and then be dispatched under a different TCB after the event completes.

BPEUXCSV environmental requirements

The requirements for the caller of BPEUXCSV are:

Authorization
Supervisor state, PSW key in which the user exit routine was originally called.

Dispatchable unit mode
TCB mode.

Save area
R13 must be pointing to a standard 72-byte save area.

Cross-memory mode
None (PASN=HASN=SASN).

AMODE
31-bit.

ASC mode
Primary.

Interrupt Status
Enabled.

Locks
None.

BPEUXCSV restrictions and limitations

This topic contains Product-sensitive Programming Interface information.

BPEUXCSV can be invoked only from within a BPE-called user exit routine. BPEUXCSV is a Product-Sensitive Programming Interface.

BPEUXCSV register information

This macro uses R0, R1, R14, and R15 as work registers. When BPEUXCSV returns control to the caller, the contents of these registers will be changed. All other registers remain unchanged.

BPEUXCSV performance implications

None.

Other macro requirements

None.

BPEUXCSV macro syntax

FUNC = CALL

The FUNC = CALL function is used to invoke a callable service from a user exit routine. The following figure shows the syntax for the CALL function.

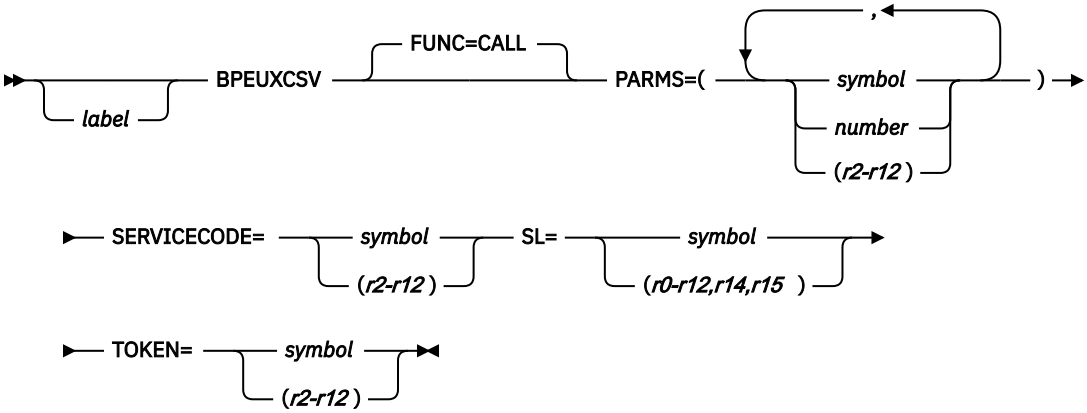


Figure 30. Syntax for BPEUXCSV macro CALL function

FUNC = DSECT

The FUNC = DSECT function is used to generate all of the following items:

- Return code symbols
- BPE callable service codes
- Parameter list DSECT for the BPEUXCSV CALL function

Syntax for BPEUXCSV macro DSECT function

Parameter descriptions

label
An optional assembler label for the macro statement.

FUNC=CALL | DSECT
An optional parameter that specifies the function of the BPEUXCSV macro. The default is CALL.

CALL
Invokes a BPE callable service from a user exit routine.

DSECT
Generates the return code symbols, BPE callable service codes, and the parameter list DSECT for the BPEUXCSV CALL function.

PARMS=(*list_of_parameters*)
A required parameter that specifies a list of subparameters (separated by commas) that are needed for the requested callable service. These subparameters are positional, and are specific to the service requested. Subparameters in this list may be in one of the following three forms.

symbol
If coded as a symbol, the value of the symbol (for example, the result of doing an `LA R0,`*symbol*) is passed as the parameter.

number
If coded as a number, the number is passed as the parameter.

(register)
If coded as a register, the content of the register is passed as the parameter. Valid registers are R2 through R12.

Examples:

- If a parameter is described as "A word in storage to receive a pointer to the returned storage," you could use one of the following coding examples.

```
BPEUXCSV PARMS=(MYWORD),...

. . .

MYWORD   DS    A             Word to receive returned ptr

        - or -

LA    2,MYWORD      Get addr of word to receive ptr
BPEUXCSV PARMS=((2)),...
```

- If a parameter is described as "The number of bytes of storage to obtain," you could use one of the following coding examples.

```
BPEUXCSV PARMS=(NUMBYTES),...

. . .

NUMBYTES EQU   1024          Number of bytes to get

        - or -

BPEUXCSV PARMS=(1024),...

        - or -

LA    5,1024
BPEUXCSV PARMS=((5)),...
```

The specific parameters and parameter order for each service are described in SERVICECODE=.

SERVICECODE=*symbol* | (*r2-r12*)
A required parameter that specifies a code that identifies the particular callable service that is being requested.

If SERVICECODE is specified as a symbol, the symbol must be an EQU symbol that is equated to the function code of the requested callable service. If SERVICECODE is specified as a register, the

register must contain the service code. For BPE-provided services, the appropriate EQU symbols are generated when you invoke BPEUXCSV FUNC = DSECT, and are specified as one of the following service codes.

BPEUXCSV_GETSTG
Get storage service.

BPEUXCSV_FREESTG
Free storage service.

BPEUXCSV_LOAD
Load module service.

BPEUXCSV_DELETE
Delete module service.

BPEUXCSV_NSCREATE
Create named storage service.

BPEUXCSV_NSRETRIEVE
Retrieve named storage service.

BPEUXCSV_NSDESTROY
Destroy named storage service.

SL=*symbol* | (*r0-r12,r14,r15*)
A required parameter that specifies an area in storage that is to be used as a service parameter list. The BPEUXCSV macro uses this storage to build the parameter list for the call to the callable service. The EQU symbol BPEUXCSV_MAXSL is generated by this macro and is equated to the size of the largest service parameter list required by BPE callable services. Ensure that area of storage you specify on the SL parameter is at least BPEUXCSV_MAXSL bytes in length when requesting any of the BPE callable services.

If SL is specified as a symbol, the symbol must be a label on the first byte of the area to be used as the service parameter list. If the SL parameter is specified as a register, the register must contain the address of the first byte of the area.

TOKEN=*symbol* | (*r2-r12*)
A required parameter that specifies the callable services token address that was passed to the user exit routine in the Standard BPE user exit parameter list field UXPL_CSTOKENP. If the TOKEN parameter is specified as a symbol, the symbol must be the label on a word of storage that contains the callable services token address. If TOKEN is specified as a register, the register must contain the callable services token address.

Return from BPEUXCSV

BPEUXCSV FUNC = CALL uses general purpose registers R0, R1, R14, and R15 as work registers. On exit from the macro, R15 is set to the return code from the BPEUXCSV macro. This return code indicates the status from the callable service request router. The possible return code values in R15 are the same for all callable service requests. R0 *might* be set to a return code for the specific callable service that was requested, depending on the value that is in R15 (see R15 return codes in the following table). The R0 return code is specific to each callable service. R1 might be set to a return value from the callable service, if applicable. See the specific service descriptions for additional information. R2 through R12 are unchanged on return from BPEUXCSV.

EQUs for the return codes in R15 are generated by BPEUXCSV FUNC = DSECT. The following table describes the possible return code values in R15 for FUNC = CALL, including the symbol, its value, and a description.

Table 185. FUNC=CALL return codes

Symbol	Value	Description
BPEUXCSV_RC_OK	X'00'	The callable service was successful.

Table 185. FUNC=CALL return codes (continued)

Symbol	Value	Description
BPEUXCSV_RC_SERV	X'04'	The specific callable service returned a non-zero return code. The return code is in the R0. Examine R0 to determine the specific reason that the request failed. The only time that the value in R0 is valid is when R15=X'04'. Otherwise, the content of R0 is unpredictable.
BPEUXCSV_RC_INVCODE	X'08'	The service code specified on SERVICECODE is invalid.
BPEUXCSV_RC_BADTOKEN	X'0C'	The callable service token passed on TOKEN is invalid.
BPEUXCSV_RC_INT	X'F4'	An internal BPE error occurred.
BPEUXCSV_RC_VERS	X'FC'	A callable services parameter list version error was encountered. The version of the parameter list generated by this macro is not valid for your current release of BPE. This is usually the result of assembling with a version of BPEUXCSV at a different level than the BPE runtime system.

BPEUXCSV get storage service

The BPEUXCSV get storage service is used to obtain virtual storage.

The BPEUXCSV get storage service is similar to the z/OS GETMAIN and STORAGE services; however, the storage obtained by the get storage service is always associated with the top-level BPE TCB (usually the job step TCB of the address space). The storage remains allocated until it is explicitly freed or until the job step TCB terminates. Therefore, you can rely on the fact that the storage stays allocated even if it is obtained under a subtask TCB which later terminates.

Service Code: BPEUXCSV_GETSTG

PARMS format:

PARMS=(*length,sp,opts*) or PARMS=(*length,sp,opts,key*)

The following are descriptions of the parameters.

length
The length of the requested storage, in bytes.

sp
The subpool of the requested storage. This must be a valid private subpool. It cannot be a common storage subpool (such as subpool 231 or 241).

opts
Options for the storage request. *opts* is a value that is the sum of several EQU values. *opts* identifies the options you have requested for the get storage service request. A BPEUXCSV FUNC = DSECT statement must be included in your module to generate the EQUs required for this function. To specify that none of the options apply, code a zero (0) for *opts*.

BPEUXCSV_GETSTG_BELOW
Include this EQU if you want LOC = BELOW (below the line) storage. If this EQU is omitted, the storage is LOC = ANY.

BPEUXCSV_GETSTG_CLEAR
Include this EQU if you want the storage to be cleared when it is returned to you. If this EQU is omitted, the storage content is unpredictable.

BPEUXCSV_GETSTG_PAGE
Include this EQU if you want the starting address of the obtained storage to be aligned on a page boundary. If this EQU is omitted, the storage is aligned on a double-word boundary.

key
The storage key of the restricted storage. *key* is an optional parameter. If coded, it indicates the storage key to be assigned to the storage returned from the get storage service. If *key* is omitted, the returned storage will be key 7 storage.

The value passed for the *key* parameter must be sixteen times the actual key value. For example, if you wanted to get key 2 storage, you would specify a value of X'20' for the *key* parameter.

Note: The *key* parameter applies *only* to subpools where KEY= applies on the z/OS GETMAIN macro (for example, subpool 229). It is ignored for all other subpools. You cannot, for example, request subpool 0 storage in a key other than 7.

Output: Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 0, the address of the obtained storage area is returned in R1. Otherwise, the content of R1 is unpredictable.

If R15 = 4 on return from this macro, R0 contains the reason code; the following table lists these return codes, including the symbol, its value, and a description.

Table 186. Get storage service return codes

Symbol	Value	Description
BPEUXCSV_GETSTG_RCSP	X'04'	An invalid or unsupported subpool was specified. Either the subpool is not supported by z/OS, or it is a common subpool.
BPEUXCSV_GETSTG_RCLV	X'08'	A zero or negative storage length was specified.
		A zero storage address was specified.
BPEUXCSV_GETSTG_RCSTG	X'0C'	The storage was unable to free the requested storage.
BPEUXCSV_GETSTG_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.
BPEUXCSV_GETSTG_RCINT	X'F4'	An internal BPE error occurred.

Examples:

- This next example shows how to get 64 bytes of storage from subpool 0. The storage is LOC = BELOW, it is aligned on a page boundary, and it is not cleared.

```
BPEUXCSV SERVICECODE=BPEUXCSV_GETSTG,                          X
      PARMS=(64,0,BPEUXCSV_GETSTG_BELOW+BPEUXCSV_GETSTG_PAGE),X
      TOKEN=UXPL_CSTOKENP,                                     X
      SL=(1)
```

- The following example shows how to get key zero storage for a length of the value in R2, from the subpool value in R3. The storage is LOC = ANY, it is not cleared, and it is double-word aligned. R4 contains the callable services token address that was passed to the user exit routine in the field UXPL_CSTOKENP.

```
BPEUXCSV SERVICECODE=BPEUXCSV_GETSTG,                      X
       PARMS=((2),(3),0, 0),                               X
       TOKEN=(4),                                          X
       SL=WORKAREA
```

BPEUXCSV free storage service

The BPEUXCSV free storage service is used to release storage that was previously obtained with the BPEUXCSV get storage service.

The free storage service is similar to the z/OS FREEMAIN service. It must be used only to release storage obtained with the get storage service. It should not be used to release storage that was obtained using any other method (such as GETMAIN).

Service Code: BPEUXCSV_FREESTG

PARMS format:

PARMS=(*address,length,sp*) or PARMS=(*address,length,sp,key*)

address
The address of the first byte of storage being released.

length
The number of bytes of the storage being released.

sp
The subpool of the storage being released. This subpool must be the same as the subpool that was specified when the storage was obtained.

key
The storage key of the storage being released. *key* is the optional parameter. If coded, it indicates the storage key of the storage being freed. If *key* is omitted, the storage must be key 7 storage.

The value passed for the *key* parameter must be sixteen times the actual key value. For example, if you were freeing key 2 storage, you would specify a value of X'20' for the *key* parameter.

Note: The *key* parameter *only* applies to subpools where KEY= applies on the z/OS FREEMAIN macro (for example, subpool 229). It is ignored for all other subpools.

Output: Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, R0 contains the reason code; the following table lists the reason codes, including the symbol, its value, and a description.

Table 187. Free storage service return codes

Symbol	Value	Description
BPEUXCSV_FREESTG_RCSP	X'04'	An invalid or unsupported subpool was specified. Either the subpool is not supported by z/OS, or it is a common subpool.
BPEUXCSV_FREESTG_RCLV	X'08'	A zero or negative storage length was specified.
BPEUXCSV_FREESTG_RCADDR	X'0C'	A zero storage address was specified.
BPEUXCSV_FREESTG_RCSTG	X'10'	The service was unable to free the requested storage.
BPEUXCSV_FREESTG_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.

Example:

This example shows how to free STGLEN bytes starting at the byte at label MYSTG in subpool 129. STGLEN is an EQU for the number of bytes to free, and MYSTG is the label on the first byte of the area to free (*not* the label on a word pointing to the area).

```
BPEUXCSV SERVICECODE=BPEUXCSV_FREESTG,                          X
      PARMS=(MYSTG,STGLEN,129),                                 X
      TOKEN=UXPL_CSTOKENP,                                      X
      SL=(1)
```

BPEUXCSV load module service

The BPEUXCSV Load Module Service is used to load a module from a library into storage.

It is similar to the z/OS LOAD service; however, the module that is loaded is always associated with the top level BPE-TCB (usually the job step TCB of the address space). The module remains allocated until it is explicitly freed or until the job step TCB terminates. Therefore, you can rely on the module remaining allocated, even if it is obtained under a subtask TCB that later terminates.

Service Code: BPEUXCSV_LOAD

PARMS format: PARMS=(*modname,dcb,opts*)

modname
Identifies an eight-character field in storage containing the name of the module to be loaded. If *modname* is coded as a symbol, the symbol must be the label on the first byte of the eight-character field. If *modname* is coded as a register, the register must contain the address of the eight-character field.

dcb
The address of an opened DCB for a partitioned data set from which to load the specified module. To use the TASKLIB, STEPLIB, or JOBLIB data sets, code 0 for this parameter.

opts
Options for the load request. *opts* is a value that is the sum of several EQU values. *opts* identifies the options you have requested for the Load Module Service request. A BPEUXCSV FUNC = DSECT statement must be included in your module to generate the EQUs required for this function. To specify that none of the options apply, code 0 for *opts*.

BPEUXCSV_LOAD_FIXED
Include this EQU if you want the module to be loaded into page-fixed storage. If this EQU is omitted, the module is loaded into pageable storage. This parameter applies only if you also specify BPEUXCSV_LOAD_GLOBAL. Otherwise, BPEUXCSV_LOAD_FIXED is ignored.

BPEUXCSV_LOAD_GLOBAL
Include this EQU if you want the module to be loaded into global (common) storage. If this EQU is omitted, it is loaded into private storage.

BPEUXCSV_LOAD_EOM
Include this EQU if you specified BPEUXCSV_LOAD_GLOBAL and you want the module to be deleted only after the address space terminates. If this EQU is omitted, the module is deleted when the top-level BPE TCB terminates. BPEUXCSV_LOAD_EOM is ignored if you did not code BPEUXCSV_LOAD_GLOBAL.

Output: If R15 = 0, the address of the loaded module is returned in R1. Otherwise, the content of R1 is unpredictable.

Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, then R0 contains the reason code; the following table lists the reason codes, including the symbol, its value, and a description.

Table 188. Load module service return codes

Symbol	Value	Description
BPEUXCSV_LOAD_RCNOTFND	X'04'	The specified module could not be found.
BPEUXCSV_LOAD_RCBLDL	X'08'	BLDL for the module failed due to an internal error.

Table 188. Load module service return codes (continued)

Symbol	Value	Description
BPEUXCSV_LOAD_RCLOAD	X'0C'	LOAD for the module failed. The module was found in the library, but LOAD returned a non-zero code.
BPEUXCSV_LOAD_RCPARM	X'F0'	An invalid number of parameters was passed to callable services request.
BPEUXCSV_LOAD_RCINT	X'F4'	An internal BPE error occurred.

Examples:

The following example shows how to load the module whose name is at the 8 bytes of storage, beginning at label MODNAME, from the default TASKLIB, JOBLIB, or STEPLIB data sets.

```
          BPEUXCSV SERVICECODE=BPEUXCSV_LOAD,                          X
                PARMS=(MODNAME,0,0),                                   X
                TOKEN=UXPL_CSTOKENP,                                   X
                SL=(1)

          . . .

MODNAME   DC    CL8'MODULE00'        Name of module to load
```

This next example shows how to load the module, whose name is at the 8 bytes of storage pointed to by R8, into global storage. The module is not deleted until the address space terminates (or until it is explicitly deleted). R2 contains the callable services token address that was passed to the user exit routine in the UXPL_CSTOKENP field. The module is loaded from the data set described by DCB MYDCB.

```
          LA    8,MODNAME           R8 = addr of name of module to load
          BPEUXCSV SERVICECODE=BPEUXCSV_LOAD,                          X
                PARMS=((8),MYDCB,BPEUXCSV_LOAD_GLOBAL+BPEUXCSV_LOAD_EOM)X
                TOKEN=(2),                                             X
                SL=PRMLIST

          . . .

MODNAME   DC    CL8'MODULE00'        Name of module to load
MYDCB     DCB   DSNAME=...
```

BPEUXCSV delete module service

The BPEUXCSV delete module service is used to delete a module that was previously loaded with the BPEUXCSV load module service.

The BPEUXCSV delete module service is similar to the z/OS DELETE service. It must be used only to delete modules obtained with the load module service. It must not be used to delete modules that were loaded using any other method (such as z/OS LOAD).

Service Code: BPEUXCSV_DELETE

PARMS format: PARMS=(*modname*)

modname
Identifies an eight character field in storage containing the name of the module to be deleted. If *modname* is coded as a symbol, the symbol must be the label on the first byte of the eight character field. If *modname* is coded as a register, the register must contain the address of the eight character field.

Output: Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, then R0 the reason code; the following table lists these reason codes, including the symbol, its value, and a description.

Table 189. Delete module service return codes

Symbol	Value	Description
BPEUXCSV_DELETE_RCDELETE	X'04'	The module that was specified could not be deleted.
BPEUXCSV_DELETE_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.
BPEUXCSV_DELETE_RCINT	X'F4'	An internal BPE error occurred.

Example:

The following example shows how to delete the module whose eight character name is in the storage pointed to by R5.

```
          LA    5,MODNAME           R5=addr of name of module to delete
          BPEUXCSV SERVICECODE=BPEUXCSV_DELETE,                         X
                PARMS=((5)),                                            X
                TOKEN=UXPL_CSTOKENP,                                    X
                SL=(1)

          . . .

MODNAME   DC    CL8'MODULE00'        Name of module to delete
```

BPEUXCSV create named storage service

The BPEUXCSV create named storage service allows you to obtain an area of storage that is associated with a 16-byte name.

In subsequent user exit routine calls (either for the same or different exit routine types), you can retrieve the named storage area address by providing the same name to the retrieve named storage service. Named storage services allow a set of user exit routines to share information but only if they agree on the same name. Typically, an initialization-type exit routine creates the named storage, and all subsequent exit routines retrieve the named storage address.

The name of the storage must be unique within the BPE address space. The named storage is obtained in subpool 0, LOC = ANY storage. The storage is cleared to zeros when it is created.

Service Code: BPEUXCSV_NSCREATE

PARMS format: PARMS=(*name,length*)

name
Identifies a 16-byte field in storage containing the name to be associated with the storage obtained. The field can contain any 16-byte value (all bytes are significant). If *name* is coded as a symbol, the symbol must be the label on the first byte of the 16-byte field. If *name* is coded as a register, the register must contain the address of the 16-byte field.

length
The number of bytes of the named storage area to obtain.

Output: If R15 = 0, the address of the named storage area obtained is returned in R1. Otherwise, the content of R1 is unpredictable.

Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, R0 contains the reason code; the following table lists these reason codes, including the symbol, its value, and a description.

Table 190. Create named storage service return codes

Symbol	Value	Description
BPEUXCSV_NSCREATE_RCDUP	X'04'	The requested storage area name is already in use.
BPEUXCSV_NSCREATE_RCLV	X'08'	A zero or negative storage length was requested.
BPEUXCSV_NSCREATE_RCNAME	X'0C'	A zero name address was specified.
BPEUXCSV_NSCREATE_RCSTG	X'10'	The service was unable to obtain the requested storage.
BPEUXCSV_NSCREATE_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.
BPEUXCSV_NSCREATE_RCINT	X'F4'	An internal BPE error occurred.

Example:

This example shows how to create a 1024-byte storage area that is associated with the 16-byte name in storage. The first byte of the named storage area is at label MYNAME.

```
          BPEUXCSV SERVICECODE=BPEUXCSV_NSCREATE,                      X
                PARMS=(MYNAME,1024),                                   X
                TOKEN=UXPL_CSTOKENP,                                   X
                SL=(1)

          . . .

MYNAME    DC    CL16'SHARED_STOR_1024'  "Name" of named storage
```

BPEUXCSV retrieve named storage service

The BPEUXCSV retrieve named storage service allows you to retrieve the address of a named area of storage that was previously created with the create named storage service.

Service Code: BPEUXCSV_NSRETRIEVE

PARMS format: PARMS=(*name*)

name
Identifies a 16-byte field in storage containing the name of the named storage area. The field can contain any 16-byte value (all bytes are significant). If *name* is coded as a symbol, the symbol must be the label on the first byte of the 16-byte field. If *name* is coded as a register, the register must contain the address of the 16-byte field.

Output: If R15 = 0, the address of the named storage area retrieved is returned in R1. Otherwise, the content of R1 is unpredictable.

Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, R0 contains the reason code; the following table lists these reason codes, including the symbol, its value, and a description.

Table 191. Retrieve named storage service return codes

Symbol	Value	Description
BPEUXCSV_NSRETRIEVE_RCNONE	X'04'	No named storage area is associated with the specified name.

Table 191. Retrieve named storage service return codes (continued)

Symbol	Value	Description
BPEUXCSV_NSRETRIEVE_RCNAME	X'08'	A zero name address was specified.
BPEUXCSV_NSRETRIEVE_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.
BPEUXCSV_NSRETRIEVE_RCINT	X'F4'	An internal BPE error occurred.

Example:

This example shows how to retrieve the address of the named storage area associated with the 16-byte name in storage at the address contained in R6.

```
          LA   6,MYNAME
          BPEUXCSV SERVICECODE=BPEUXCSV_NSRETRIEVE,                    X
                PARMS=((6)),                                           X
                TOKEN=UXPL_CSTOKENP,                                   X
                SL=(1)

          . . .

MYNAME    DC    CL16'SHARED_STOR_1024'  "Name" of named storage
```

BPEUXCSV destroy named storage service

The BPEUXCSV destroy named storage service is used to delete a previously created named storage area. No other user exit routine should access this storage after you destroy it.

Service Code: BPEUXCSV_NSDESTROY

PARMS format: PARMS=(*name*)

name
Identifies a 16-byte field in storage containing the name of the named storage area. The field can contain any 16-byte value (all bytes are significant). If *name* is coded as a symbol, the symbol must be the label on the first byte of the 16-byte field. If *name* is coded as a register, the register must contain the address of the 16-byte field.

Output: Return code EQUs are generated by BPEUXCSV FUNC = DSECT. If R15 = 4 on return from this macro, R0 contains the reason code; the following table lists these reason codes, including the symbol, its value, and a description.

Table 192. Destroy named storage service return codes

Symbol	Value	Description
BPEUXCSV_NSDESTROY_RCNONE	X'04'	No named storage area is associated with the specified name.
BPEUXCSV_NSDESTROY_RCNAME	X'08'	A zero name address was specified.
BPEUXCSV_NSDESTROY_RCPARM	X'F0'	An invalid number of parameters was passed to the callable services request.
BPEUXCSV_NSDESTROY_RCINT	X'F4'	An internal BPE error occurred.

Example:

The following example shows how to destroy the named storage area associated with the 16-byte name in storage whose first byte is at label NSNAME.

```
          BPEUXCSV SERVICECODE=BPEUXCSV_NSDESTROY,                    X
                PARMS=(NSNAME),                                       X
                TOKEN=UXPL_CSTOKENP,                                  X
                SL=(1)

          . . .

NSNAME    DC    XL16'01C1C2C300000000F1F2F3F4006D2748'  Binary names OK
```

BPE callable service example: Sharing data among exit routines

As an example of the use of callable services, consider the case where you have a set of user exit routines of varying types that all need to share some common information.

For this example, assume that the following three types of exit routines are being used:

- An initialization exit routine that gets control when the address space is first started. Assume that this exit routine runs before any mainline processing is done (so you can be sure that the other two exit routines will not be called until the initialization exit routine has returned).
- A processing exit routine that gets control whenever a particular event occurs in the address space that needs user exit routine provided information.
- A termination exit routine that gets control when the address space is ending.

Important: These user exit routines are presented here for example purposes only. These examples should not be assumed to be usable exit routines.

Subsections:

- "Sample initialization exit routine" on page 504
- "Sample processing exit routine" on page 505
- "Sample termination exit routine" on page 506

Sample initialization exit routine

The initialization exit routine uses the create named storage service to obtain a 16-byte area of storage with the name ZZZ_EXIT_AREA. The storage is mapped by the following DSECT (which is assumed to be available in all of the modules).

```
ZZZ_EXIT_AREA    DSECT ,
ZZZ_TABLE_NAME   DS    CL8         Name of table module
ZZZ_TABLE_ADDR   DS    A           Address of table module
                 DS    F           Available
ZZZ_EXIT_AREA_L  EQU   *-ZZZ_EXIT_AREA
```

The initialization exit routine then uses the load module service to load a module named ZZZUXTB0 (a table that is needed in this example to pass information to the other user exit routines). The initialization exit routine stores the name of the table module in the named storage area field ZZZ_TABLE_NAME, and the address of the loaded table in field ZZZ_TABLE_ADDR. A routine using a table may not be required for your application.

A sample initialization exit routine that performs these functions is shown in the following example. The code shown in the following examples is mainline path only. For simplicity, error paths and exception handling code are not shown.

```
INITEXIT CSECT ,
INITEXIT AMODE 31
INITEXIT RMODE ANY
         STM   14,12,12(13)           Save caller's registers
         LR    12,15                  Move module entry pt to R12
         USING INITEXIT,12            Address module base register
```

```
         L     13,8(,13)               Chain to 2nd provided save area
         LR    11,1                    Move exit parmlist to R11
         USING BPEUXPL,11              Address std BPE user exit PL
         L     10,UXPL_DYNAMICWAP      Get 512-byte dynamic storage ptr
         USING DYNSTG,10               Address module's dynamic storage
         BPEUXCSV SERVICECODE=BPEUXCSV_NSCREATE,  Create named stg     X
               PARMS=(NSNAME,ZZZ_EXIT_AREA_L),     for the exits       X
               TOKEN=UXPL_CSTOKENP,                                    X
               SL=UXCSVPL
         LTR   15,15                   Did NSCreate work?
         BNZ   ERROR1                  No, go handle error

         LR    9,1                     Yes, named storage ptr to R9
         USING ZZZ_EXIT_AREA,9         Address using "ZZZ" DSECT
         MVC   ZZZ_TABLE_NAME,TBLNAME  Set name of table module
         BPEUXCSV SERVICECODE=BPEUXCSV_LOAD,      Load the table       X
               PARMS=(TBLNAME,0,0),                module for the      X
               TOKEN=UXPL_CSTOKENP,                exits               X
               SL=UXCSVPL
         LTR   15,15                   Did LOAD work?
         BNZ   ERROR2                  No, go handle error
         ST    1,ZZZ_TABLE_ADDR        Yes, save table ptr in named stg

         . . .                         Do any other init exit functions

         XR    15,15                   Set zero return code
         L     13,4(,13)               Back up to caller's save area
         L     14,12(,13)              Restore caller's R14
         LM    0,12,20(13)             Restore caller's R0-R12
         BR    14                      Return to caller
         DROP  9,10,11,12              Release USING registers

NSNAME   DC    CL16'ZZZ_EXIT_AREA   '  Const for named storage
TBLNAME  DC    CL8'ZZZUXTB0'           Const for table module name
         LTORG ,

DYNSTG   DSECT ,                       Dynamic storage DSECT
UXCSVPL  DS    XL(BPEUXCSV_MAXSL)      Space for BPEUXCSV parmlist
         . . .                         Other dynamic storage fields
         BPEUXPL  FUNC=DSECT           Include user exit parmlist
         BPEUXCSV FUNC=DSECT           Include BPEUXCSV symbols
         END
```

Sample processing exit routine

The processing exit routine obtains the address of the table module that was loaded by the initialization exit routine. For optimum performance, the processing exit routine uses the first word of the static work area that BPE passes to save the address of the shared storage area.

On entry, the processing exit routine checks this word of storage. If this word is non-zero, the processing routine uses this address as the shared storage area pointer. If the first word is zero, the processing exit routine invokes the named storage retrieve service to get the address of the shared storage. The processing exit routine then saves the address in the static storage area. This technique minimizes the number of BPE requests for callable services that this exit routine must make (because it needs to do the retrieve only once; on subsequent calls, the address of the shared storage area is available in the static work area).

A sample processing exit routine that performs these functions is shown in the following example.

```
PROCEXIT CSECT ,
PROCEXIT AMODE 31
PROCEXIT RMODE ANY
         STM   14,12,12(13)            Save caller's registers
         LR    12,15                   Move module entry pt to R12
         USING PROCEXIT,12             Address module base register
         L     13,8(,13)               Chain to 2nd provided save area
         LR    11,1                    Move exit parmlist to R11
         USING BPEUXPL,11              Address std BPE user exit PL
         L     10,UXPL_DYNAMICWAP      Get 512-byte dynamic storage ptr
         USING DYNSTG,10               Address module's dynamic storage
         L     9,UXPL_STATICWAP        Get 256-byte static storage ptr
         ICM   8,15,0(9)               Is shared stg ptr set?
         BNZ   GOTSHRD                 Yes, continue
         BPEUXCSV SERVICECODE=BPEUXCSV_NSRETRIEVE, Get named stg addr  X
```

```
               PARMS=(NSNAME),                                         X
               TOKEN=UXPL_CSTOKENP,                                    X
               SL=UXCSVPL
         LTR   15,15                   Did NSRetrieve work?
         BNZ   ERROR1                  No, go handle error
         LR    8,1                     Yes, set shrd stg ptr in R8
         ST    8,0(,9)                 Save in static stg for next time
GOTSHRD  DS    0H
         USING ZZZ_EXIT_AREA,8         Address using "ZZZ" DSECT
         L     7,ZZZ_TABLE_ADDR        Get table address

         . . .                         Do process exit functions
         XR    15,15                   Set zero return code
         L     13,4(,13)               Back up to caller's save area
         L     14,12(,13)              Restore caller's R14
         LM    0,12,20(13)             Restore caller's R0-R12
         BR    14                      Return to caller

         DROP  8,10,11,12              Release USING registers
NSNAME   DC    CL16'ZZZ_EXIT_AREA   '  Const for named storage

         LTORG ,

DYNSTG   DSECT ,                       Dynamic storage DSECT
UXCSVPL  DS    XL(BPEUXCSV_MAXSL)        Space for BPEUXCSV parmlist
         . . .                           Other dynamic storage fields
         BPEUXPL  FUNC=DSECT           Include user exit parmlist
         BPEUXCSV FUNC=DSECT           Include BPEUXCSV symbols
         END
```

Sample termination exit routine

The termination exit routine locates the shared storage area, deletes the loaded table module using the name that was saved in the shared storage area, and then destroys the shared area.

A sample termination exit routine that performs these functions is shown in the following example.

```
TERMEXIT CSECT ,
TERMEXIT AMODE 31
TERMEXIT RMODE ANY
         STM   14,12,12(13)            Save caller's registers
         LR    12,15                   Move module entry pt to R12
         USING TERMEXIT,12             Address module base register
         L     13,8(,13)               Chain to 2nd provided save area
         LR    11,1                    Move exit parmlist to R11
         USING BPEUXPL,11              Address std BPE user exit PL
         L     10,UXPL_DYNAMICWAP      Get 512-byte dynamic storage ptr
         USING DYNSTG,10               Address module's dynamic storage
         BPEUXCSV SERVICECODE=BPEUXCSV_NSRETRIEVE, Get named stg addr  X
               PARMS=(NSNAME),                                         X
               TOKEN=UXPL_CSTOKENP,                                    X
               SL=UXCSVPL
         LTR   15,15                   Did NSRetrieve work?
         BNZ   ERROR1                  No, go handle error
         LR    8,1                     Yes, set shrd stg ptr in R8
         USING ZZZ_EXIT_AREA,8         Address using "ZZZ" DSECT
         BPEUXCSV SERVICECODE=BPEUXCSV_DELETE,     Delete table        X
               PARMS=(ZZZ_TABLE_NAME),              module             X
               TOKEN=UXPL_CSTOKENP,                                    X
               SL=UXCSVPL
         LTR   15,15                   Did DELETE work?
         BNZ   ERROR2                  No, go handle error
         BPEUXCSV SERVICECODE=BPEUXCSV_NSDESTROY,  Destroy named stg   X
               PARMS=(NSNAME),                                         X
               TOKEN=UXPL_CSTOKENP,                                    X
               SL=UXCSVPL
         DROP  8                       R8 no longer is "ZZZ" area
         LTR   15,15                   Did NSDestroy work?
         BNZ   ERROR3                  No, go handle error
         . . .                         Do other term exit functions

         XR    15,15                   Set zero return code
         L     13,4(,13)               Back up to caller's save area
         L     14,12(,13)              Restore caller's R14
         LM    0,12,20(13)             Restore caller's R0-R12
         BR    14                      Return to caller
         DROP  10,11,12                Release USING registers
```

```
NSNAME   DC    CL16'ZZZ_EXIT_AREA   '  Const for named storage

         LTORG ,
DYNSTG   DSECT ,                       Dynamic storage DSECT
UXCSVPL  DS    XL(BPEUXCSV_MAXSL)        Space for BPEUXCSV parmlist
         . . .                           Other dynamic storage fields

         BPEUXPL  FUNC=DSECT           Include user exit parmlist
         BPEUXCSV FUNC=DSECT           Include BPEUXCSV symbols
         END
```

Chapter 6. Base Primitive Environment customization exit routines

BPE customization user exit routines enable you to customize and monitor address spaces built on the Base Primitive Environment.

BPE-defined user exit routine types are available to all IMS component address spaces that run with BPE. You write these exit routines. No sample exit routines are provided. The BPE user exit routines are given control in the address space in an authorized state.

Recommendation: Write BPE user exit routines in assembler, not in a high level language. BPE does not support exit routines that run under Language Environment for z/OS. If you write an exit routine in a high level language, and that routine runs in the Language Environment for z/OS, you might have abends or performance problems. Language Environment for z/OS is designed for applications running in key 8, problem program state. BPE user exit routines execute in key 7 supervisor state.

BPE Initialization-Termination user-supplied exit routine

The BPE Initialization-Termination user-supplied exit routine is called during BPE initialization and normal BPE termination.

BPE user exit routines enable you to customize and monitor address spaces built on the Base Primitive Environment. BPE-defined user exit routine types are available to all IMS component address spaces that run with BPE. You write these exit routines. No sample exit routines are provided. The BPE user exit routines are given control in the address space in an authorized state.

Recommendation: Write BPE user exit routines in assembler, not in a high level language. BPE does not support exit routines that run under Language Environment for z/OS. If you write an exit routine in a high level language, and that routine runs in the Language Environment for z/OS, you might have abends or performance problems. Language Environment for z/OS is designed for applications running in key 8, problem program state. BPE user exit routines execute in key 7 supervisor state.

This topic contains Product-sensitive Programming Interface information.

Subsection:

- "About this routine" on page 509

About this routine

The Init-Term exit routine is not called during BPE abnormal termination. This exit routine is optional.

The Init-Term exit routine is defined as TYPE = INITTERM, COMP=BPE in the EXITDEF statement in the BPE user exit PROCLIB member pointed to by the EXITMBR statement for the BPE exit routines. You can specify one or more user exit routines of this type. When the init-term exit point is reached, the exit routines are driven in the order they are specified by the EXITS= keyword.

Recommendation: Write the Init-Term exit routine so that it is reentrant. The Init-Term exit routine is invoked AMODE 31.

Contents of registers on entry

Register
Contents

1
Address of Standard BPE user exit parameter list (mapped by the BPEUXPL macro).

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Always set this to zero.

All other registers must be restored.

BPE initialization and termination parameter list

On entry to the Init-Term exit routine, R1 points to a Standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the Init-Term user exit routine parameter lists (mapped by the BPEITXP macro). The following table provides the following information about the BPE Init-Term user exit routine parameters:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Field name	Offset	Length	Field usage	Description
BPEITXP	X'00'	N/A	N/A	DSECT label for the BPE init-term exit parameter list
BPEITXP_VERSION	X'00'	X'04'	Input	Parameter list version number (X'00000001')
BPEITXP_FUNC	X'04'	X'04'	Input	Function code **1** BPE Initialization (BPEITXP_FUNC_INIT) **2** BPE Termination (BPEITXP_FUNC_TERM)

Related reference
“BPE user-supplied exit routine interfaces and services” on page 485

BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

BPE Statistics user-supplied exit routine

The BPE Statistics user-supplied exit routine is called at regular intervals during the life of a BPE address space, and a final time at normal address shutdown, to gather address-space related statistics.

Recommendation: Write BPE user exit routines in assembler, not in a high level language. BPE does not support exit routines that run under Language Environment for z/OS. If you write an exit routine in a high level language, and that routine runs in the Language Environment for z/OS, you might have abends or performance problems. Language Environment for z/OS is designed for applications running in key 8, problem program state. BPE user exit routines execute in key 7 supervisor state.

Subsection:

- "About this routine" on page 511

About this routine

This topic contains Diagnosis, Modification, and Tuning information.

The BPE Statistics user exit routine enables you, at regular intervals, to gather statistics related to an IMS component that is running with a BPE address space. The exit routine is also called a final time during normal shutdown of the address space. The BPE Statistics user exit routine is optional.

The statistics exit routine is called on a time-driven basis. The interval between successive statistics exit routine calls is specified on the STATINTV parameter in the BPE configuration PROCLIB member. The exit routine is first called soon after BPE initialization completes. Subsequent calls occur every STATINTV seconds after the previous call returns.

The BPE statistics exit routine is also called one final time during normal address space shutdown processing. When it is called for normal shutdown, the function code passed in the BPESTXP parameter list will be BPESTXP_FUNC_FINALSTATS (2), indicating that this is the final statistics exit routine call.

The BPE Statistics user exit routine is defined as TYPE = STATS, COMP=BPE in the EXITDEF statement in the BPE user exit PROCLIB member. You can specify one or more user exit routines of this type. When this exit routine is invoked, all routines of this type are driven in the order specified by the EXITS= keyword.

Important: All statistics passed to the BPE Statistics user exit routine are considered Diagnosis, Modification, or Tuning Information.

Recommendation: Write the BPE Statistics exit routine so that it is reentrant. It is invoked AMODE 31.

Contents of registers on entry

Register
Contents

1
Address of Standard BPE user exit parameter list (mapped by the BPEUXPL macro).

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Always set this to zero.

All other registers must be restored.

BPE statistics exit routine parameter list

On entry to the Statistics exit routine, R1 points to a Standard BPE user exit parameter list. Field UXPL_EXITPLP in the Standard BPE user exit parameter list contains the address of the BPE Statistics user exit routine parameter list (mapped by the BPESTXP macro). The following table provides the following information about the Statistics user exit routine parameters:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Table 193. BPE statistics user-supplied exit routine parameter list

Field name	Offset	Length	Field usage	Description
BPESTXP	X'00'	N/A	N/A	DSECT label for the BPE statistics exit parameter list
BPESTXP_VERSION	X'00'	X'04'	Input	Parameter list version number (00000001)
BPESTXP_FUNC	X'04'	X'04'	Input	Function code **1** Statistics (BPESTXP_FUNC_STATS) **2** Final statistics (BPESTXP_FUNC_FINALSTATS)
BPESTXP_BPESTATS_PTR	X'08'	X'04'	Input	Address of BPE system statistics area header. This header points to detailed BPE system statistics. All of the BPE statistics areas are mapped by macro BPESSTA.
BPESTXP_COMPSTATS_PTR	X'0C'	X'04'	Input	Address of the IMS component statistics area, or zero if none. An IMS component that runs with BPE has the ability to define its own statistics area, to be passed along with the BPE statistics area when the BPE statistics exit is called. However, not all IMS components provide their own statistics in that manner. If a component does not provide statistics, this field in the BPESTXP parameter list is zero.

Related reference

“BPE user-supplied exit routine interfaces and services” on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

“CQS statistics available through the BPE statistics user-supplied exit” on page 572
You can use the BPE Statistics user exit to gather both BPE and CQS statistics.

“CSL ODBM statistics available through BPE statistics user exit” on page 586
The BPE Statistics user exit can be used to gather both BPE and ODBM statistics.

“CSL OM statistics available through BPE statistics user exit” on page 603
The BPE Statistics user exit can be used to gather both BPE and OM statistics.

“CSL SCI statistics available through BPE statistics user exit” on page 620
The BPE Statistics user exit can be used to gather both BPE and SCI statistics.

“CSL RM statistics available through BPE statistics user exit” on page 611
The BPE Statistics user exit can be used to gather both BPE and RM statistics.

“DBRC statistics” on page 543

You can use the BPE Statistics user-supplied exit to gather both BPE and DBRC statistics.

BPE system statistics area

The BPE system statistics area contains statistics on system resources managed by BPE.

The BPE system statistics area contains statistics on the following system resources managed by BPE.

- TCBs
- Control block services
- AWE servers
- Storage services

The field BPESTXP_BPESTATS_PTR in the BPE statistics exit parameter list points to this area. The following figure shows the structure of the BPE system statistics area.

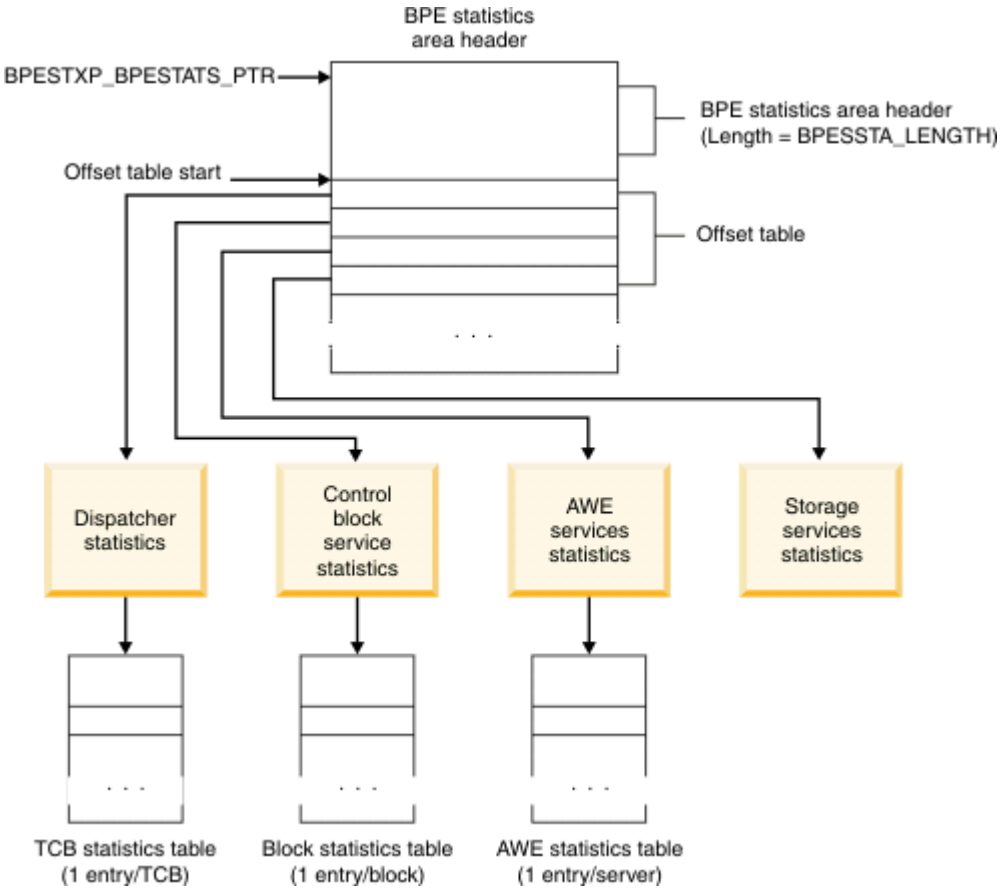


Figure 31. BPE system statistics area structure

The BPE system statistics area begins with the BPESSTA header. The header contains general information about the BPE address space and the IMS component running in it. The offset table appears immediately after the header (BPESSTA + SSTA_LENGTH). Each area for which statistics are reported is assigned a fixed slot within this table. Each slot contains the offset to the particular area's statistics block from the start of the offset table. Each area may have one or more blocks for the statistics pertaining to the area.

All "pointers" among the BPE system statistics area blocks are really offsets, not addresses. Having offsets allows statistics to be written to a log or other data set, where the original block addresses are no longer meaningful. All offsets are relative to the beginning of the DSECT in which the offset field resides.

The total length of the BPE statistics area is not fixed (static). The length depends on the resource definitions and number of active resources in the system. Many of the area blocks contain entries for each resource type.

Recommendation: Always use the lengths passed in the area fields to refer to the length of a particular statistics area. Do not use lengths generated as EQUs (assembler equates) at assembly time. Using the passed lengths ensures that your exit routine code works correctly, even if the format of the statistics areas changes in the future.

Unless otherwise indicated, the following statements are true:

- All statistics in the various BPE statistics area topics are cumulative since the start of the address space.
- Count fields are 32-bit unsigned numbers.
- Time-related double-word fields are in STCK units (bit 51 = 1 microsecond, where bit 51 means the 51st bit from the left, or the 12th bit from the right)
- Statistics are gathered without serialization for performance reasons. As a result, the statistics might not be completely consistent with each other. For example, two related statistics might be updated at different times. View the statistics as aggregate indications of the system performance, not as exact values.

The following table provides the following information about the fields in the BPE system statistics area:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Table 194. BPE system statistics area

Field name	Offset	Length	Field usage	Description
BPESSTA	X'00'	N/A	N/A	DSECT label for the BPE system statistics area.
SSTA_ID	X'00'	X'08'	Input	Eye catcher ("BPESSTA ").
SSTA_LENGTH	X'08'	X'04'	Input	BPESSTA header section length (SSTA_END minus BPESSTA). The offset table starts immediately after the BPESSTA header (BPESSTA + SSTA_LENGTH).
SSTA_VER	X'0C'	X'04'	Input	BPESSTA header version number within a BPE release. The current version is X'00000001' (SSTA_VER_1).
SSTA_BPEVER	X'10'	X'03'	Input	BPE version number.
SSTA_CONDSRB	X'13'	X'01'	Input	The value of the CONDSRB BPE configuration PROCLIB member parameter in effect for this address space. This field is one of the following values: SSTA_CONDSRB_NEVER (1) CONDSRB(NEVER) SSTA_CONDSRB_COND (2) CONDSRB(COND) SSTA_CONDSRB_ALWAYS (3) CONDSRB(ALWAYS)
SSTA_OFSTTBLLEN	X'14'	X'04'	Input	Length of offset table.
SSTA_UTYPE	X'18'	X'04'	Input	IMS component type.

Table 194. BPE system statistics area (continued)

Field name	Offset	Length	Field usage	Description
SSTA_UVERSION	X'1C'	X'03'	Input	IMS component version number.
	X'1F'	X'01'	Input	Reserved.
SSTA_USYSNAME	X'20'	X'08'	Input	IMS component system name.
SSTA_JOBNAME	X'28'	X'08'	Input	Jobname of address space for which this record was created.
SSTA_STARTSTCK	X'30'	X'08'	Input	STCK at BPE start (STCK when BPE job step TCB was created).
SSTA_STCK	X'38'	X'08'	Input	STCK when this record was created.
STCK_LDTO	X'40'	X'08'	Input	Local time-date offset from field CVTLDTO in the CVT (the amount to add to a UTC STCK to get local time STCK, in STCK units).
SSTA_CPUID	X'48'	X'08'	Input	CPU ID from STIDP instruct.
SSTA_UPRODNUM	X'50'	X'08'	Input	Product number (comp-ID) of IMS component, in the form of *nnnn-nnn*. This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_OSNAME	X'58'	X'08'	Input	Operating system name (from field CVTSNAME in the ECVT). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_OSPNAME	X'60'	X'10'	Input	Operating system product name (from field ECVTPNAM in the ECVT). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_OSPVER	X'70'	X'02'	Input	Operating system version, in EBCDIC (from field ECVTPVER in the ECVT). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_OSPREL	X'72'	X'02'	Input	Operating system release, in EBCDIC (from field ECVTPVER in the ECVT). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_OSPMOD	X'74'	X'02'	Input	Operating system modification level, in EBCDIC (from field ECVTPREL in the ECVT). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_SYSCLONE	X'76'	X'02'	Input	SYSCLONE value (from field ECVTCLON in the ECVT). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.

Table 194. BPE system statistics area (continued)

Field name	Offset	Length	Field usage	Description
SSTA_TOTALLEN	X'78'	X'04'	Input	Total length of all statistics areas. The total length is the number of bytes from the start of the BPESSTA to the last byte of statistics data. You can use this field if you are copying the statistics data to another location (for example, to a data set) to determine the length of the data to copy. This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTA_#CPS	X'7C'	X'04'	Input	Number of online standard CPs. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTA_#ZAAPS	X'80'	X'04'	Input	Number of online zAAPs. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTA_#ZIIPS	X'84'	X'04'	Input	Number of online zIIPs. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTA_FIELDFLAGS	X'88'	X'08'	Input	Field indicator flags. New fields added to the BPE statistics area in the future have a bit assigned to them in these indicator flags to show that they are present. Code that wants to look at the new fields can test their assigned bits to determine whether they are present in the statistics area passed by BPE. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater. A field associated with a particular flag is annotated in the field's comments with the end of the flag's name in parentheses. For example: "(LVL001)" for a field added with flag SSTA_FLDF1_LVL001.
SSTA_FLDF1	X'88'	X'01'	Input	Field indicator flag byte 1. Currently defined bits in this byte are: **SSTA_FLDF1_LVL001 (X'80')** The following fields are present (LVL001): • SSTADS_#FORCEDISP • SSTAAW_IDX • SSTAAW_MAXCONSECAWE • SSTAAW_MAXCAWECT • SSTAAW_IDLEWORKCT • SSTAAW_MAXAWESTCK • SSTAAW_MAXAWEPROC

The BPE statistics offset table is immediately after the BPE statistics header (BPESSTA + SSTA_LENGTH). The offset table contains offsets to the various statistics blocks in the area.

Attention: The values in the table are offsets from the start of the offset table, not from BPESSTA. You must use the offset table to locate the different statistics sections to allow for changes to the lengths of these sections.

The following example demonstrates how to locate the dispatcher statistics area, assuming that R2 points to the BPESSTA header:

```
          USING     BPESSTA,R2              Address SSTA header
          LR        R3,R2                   Copy SSTA header addr
          AL        R3,SSTA_LENGTH          Add length to get ofst tble addr
          USING     SSTA_OFSTTBL,R3         Address offset table
          ICM       R4,15,SSTA_OFST_DISP    Any dispatcher section?
          BZ        NODSP                   No, can't access it
          ALR       R4,R3                   Add ofst tbl start to get addr
          USING     SSTADS,R4               Address dispatcher section
          . . .
NODSP     DS        OH                      To here if no disp sect present
```

Check offset fields for values of zero before using them. A zero offset field means that the particular statistics block is not present in the area.

The following table provides the following information about the fields in the BPE statistics offset table:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Table 195. BPE statistics offset table

Field name	Offset	Length	Field usage	Description
SSTA_OFSTTBL	X'00'	N/A	N/A	DSECT label for the BPE statistics offset table.
SSTA_OFST_DISP	X'00'	X'04'	Input	Offset to dispatcher statistics.
SSTA_OFST_CBS	X'04'	X'04'	Input	Offset to control block services statistics.
SSTA_OFST_AWE	X'08'	X'04'	Input	Offset to AWE statistics.
SSTA_OFST_STG	X'0C'	X'04'	Input	Offset to general storage statistics. This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.

The following table provides the following information about the fields in the BPE dispatcher statistics area:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Table 196. BPE dispatcher statistics area

Field name	Offset	Length	Field usage	Description
SSTADS	X'00'	N/A	N/A	DSECT label for BPE dispatcher statistics area.
SSTADS_ID	X'00'	X'04'	Input	Dispatcher section eye catcher ("DISP").
SSTADS_LENGTH	X'04'	X'04'	Input	Length of dispatcher section (includes TCB statistics table).
SSTADS_VERSION	X'08'	X'04'	Input	Dispatcher statistics version number. The current version is X'00000001' (SSTADS_VER_1).
SSTADS_TBLOFST	X'0C'	X'04'	Input	Offset from SSTADS to the first TCB statistics table entry.
SSTADS_NUMENT	X'10'	X'02'	Input	Number of TCB statistics table entries.
SSTADS_ENTLEN	X'12'	X'02'	Input	Length of each TCB statistics entry.
SSTADS_THD#	X'14'	X'04'	Input	Global number of thread starts.
SSTADS_DISP#	X'18'	X'04'	Input	Global number of dispatches.
	X'1C'	X'04'	Input	Reserved.
SSTADS_TREALTM	X'20'	X'08'	Input	Real time (wall clock time) that the BPE address space has been running (in STCK units). This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTADS_TBPETM	X'28'	X'08'	Input	Total time that all BPE-managed TCBs have been dispatched by the BPE dispatcher (in STCK units). This time is cumulative and includes time for TCBs that have terminated, as well as those that are still active. This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTADS_TCPUTM	X'30'	X'08'	Input	Total CPU time used by all BPE-managed TCBs (in STCK units). This time is cumulative and includes time for TCBs that have terminated, as well as those that are still active. This field is present only if the BPE version (SSTA_BPEVER) is X'010400' or greater.
SSTADS_TBPETMSRB	X'38'	X'08'	Input	Total time that all BPE-managed SRBs have been dispatched by the BPE dispatcher (in STCK units). This time is cumulative and includes time for SRBs that have terminated, as well as those that are still active. Note that this value might not have been updated with CPU time from currently executing SRBs. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_TCPUTMSRB	X'40'	X'08'	Input	Total CPU time used on a standard CP by all BPE-managed SRBs (in STCK units). This time is cumulative and includes time for SRBs that have terminated, as well as those that are still active. This value might not be updated with CPU time from currently executing SRBs. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.

Table 196. BPE dispatcher statistics area (continued)

Field name	Offset	Length	Field usage	Description
SSTADS_TCPUTMSRBZ	X'48'	X'08'	Input	Total CPU time used on a specialty engine (zIIP or zAAP) by all BPE-managed SRBs (in STCK units). This time is cumulative and includes time for SRBs that have terminated, as well as those that are still active. This value might not be updated with CPU time from currently executing SRBs. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_ZCPUTCB	X'50'	X'08'	Input	Address space TCB CPU time in ASCB field ASCBEJST at the time the BPE statistics exit is called. The ASCBEJST field might not be updated with CPU time from currently executing units of work. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_ZCPUSRB	X'58'	X'08'	Input	Address space SRB CPU time in ASCB field ASCBSRBT at the time the BPE statistics exit is called. The ASCBSRBT field might not be updated with CPU time from currently executing units of work. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_ZCPUENCLZ	X'60'	X'08'	Input	Address space enclave zIIP time in ASSB field ASSB_ZIIP_ENCT at the time the BPE statistics exit is called. ASSB_ZIIP_ENCT might not be updated with CPU time from currently executing units of work. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_ZCPUNENCZ	X'68'	X'08'	Input	Address space zIIP time (not including enclave time) in ASSB field ASSB_TIME_ON_ZIIP at the time the BPE statistics exit is called. Note that ASSB_TIME_ON_ZIIP might not have been updated with CPU time from currently executing units of work. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_ZCPUCPZ	X'70'	X'08'	Input	Address space zIIP time on standard CP in ASSB field ASSB_TIME_ZIIP_ON_CP at the time the BPE statistics exit is called. The ASSB_TIME_ZIIP_ON_CP field might not be updated with CPU time from currently executing units of work. The value is approximate. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.

The following table provides the following information about the BPE TCB statistics table entry:

- The field name
- The offset
- The length

- The field usage
- A description of the field

BPE supports SRB-mode dispatchable units (DUs) in addition to TCB-mode DUs. Both TCB and SRB DU types have entries in the BPE TCB statistics table.

Each active BPE-managed dispatchable unit (TCB or SRB) in the address space has one table entry. In the case of DU types that support multiple DUs, each instance of a DU has one entry.

Important: The BPE and the IMS component running on the BPE can define a DU type with the same name. Use the SSTADS_F1_SYS flag to differentiate between the DUs in this case.

Table 197. BPE TCB statistics table entry

Field name	Offset	Length	Field usage	Description
SSTADS_TTE	X'00'	N/A	N/A	DSECT label for BPE TCB statistics table entry.
SSTADS_TYPE	X'00'	X'04'	Input	Dispatchable unit (DU) type.
SSTADS_FLG1	X'04'	X'01'	Input	DDB flag 1 (unlabeled bits are reserved by IBM). **SSTADS_F1_POOL (X'08')** Dispatchable unit is a pool-type DU. **SSTADS_F1_SYS (X'10')** DU is BPE-defined. **SSTADS_F1_ISSRB (X'01')** DU is in SRB mode.
SSTADS_FLG2	X'05'	X'01'	Input	DDB flag 2 (unlabeled bits are reserved by IBM).
SSTADS_IDX	X'06'	X'01'	Input	DU index number.
SSTADS_INUM	X'07'	X'01'	Input	DU instance number.
SSTADS_BPE_TCBTKN	X'08'	X'08'	Input	TCB token (unique value identifying this DU).
SSTADS_#THDCR	X'10'	X'04'	Input	Number of thread creates.
SSTADS_#THDDL	X'14'	X'04'	Input	Number of thread deletes.
SSTADS_#THDSTART	X'18'	X'04'	Input	Number of thread starts.
SSTADS_#THDDISP	X'1C'	X'04'	Input	Number of thread dispatches.
SSTADS_#SUSP	X'20'	X'04'	Input	Number of suspends.
SSTADS_#SUSPBKO	X'24'	X'04'	Input	Number of backed-out suspends.
SSTADS_REALTIME	X'28'	X'08'	Input	Wall-clock time the DU has been running (in STCK units).
SSTADS_BPETIME	X'30'	X'08'	Input	Time the DU has been dispatched by BPE dispatcher (in STCK units).
SSTADS_CPUTIME	X'38'	X'08'	Input	CPU time on a standard CP for this DU (in STCK units).
SSTADS_CPUTIMEZ	X'40'	X'08'	Input	CPU time on a specialty engine (zIIP or zAAP) for this DU (in STCK units). This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.

Table 197. BPE TCB statistics table entry (continued)

Field name	Offset	Length	Field usage	Description
SSTADS_#TCBSW	X'48'	X'04'	Input	Cumulative count of number of BPETCBSW (TCB/SRB switch) calls from this DU. This field exists only if the BPE version field (SSTA_BPEVER) is X'010900' or greater.
SSTADS_QLENCT	X'4C'	X'04'	Input	Cumulative count of the number of threads on the posted and ready queues at thread dispatch. The value is the number of threads remaining. It does not include the thread being dispatched. This value, divided by the number of dispatches, equals the average queue length behind the thread being dispatched. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_QLENMAX	X'50'	X'04'	Input	Maximum number of threads on the posted and ready queues at thread dispatch since the last call to the BPE statistics exit. This field is present only if the BPE version (SSTA_BPEVER) is X'010900' or greater.
SSTADS_#FORCEDISP	X'54'	X'04'	Input	Number of times a thread gives up control on the TCB via BPEWAIT FORCEDISP=YES and there are other ready threads (LVL001).
	X'58'	X'08'	Input	Reserved for future use (LVL001).

Note:

- Entries in this table can remain unused if a TCB terminates after the statistics module computes the number of entries in the table, but before all of the statistics are captured.
- The entries in this table might not be in the same order every time the statistics area is generated. You must use the 8-byte TCB token to associate entries if you are computing entries from two different statistics exit calls.
- The number of entries in the statistics table might change from one BPE statistics user exit routine call to the next. TCBs are dynamic and might be created and destroyed as part of normal processing. The TCB statistics table represents the TCBs currently active at the time the statistics exit routine was called.

The following table provides the following information about the fields in the BPE control block services (CBS) statistics area:

- The field name
- The offset
- The length
- The field usage
- A description of the field

The control blocks services area contains a header with global statistics and information, followed by a table with one entry for each CBS-defined block type in the system.

Table 198. BPE control block services statistics area

Field name	Offset	Length	Field usage	Description
SSTACB	X'00'	N/A	N/A	DSECT label for BPE control block services statistics area.
SSTACB_ID	X'00'	X'04'	Input	Control block services section eye catcher ("CBS").
SSTACB_LENGTH	X'04'	X'04'	Input	Length of CBS section (includes block statistics table).
SSTACB_VERSION	X'08'	X'04'	Input	Control block services statistics version number. The current version is X'00000001' (SSTACB_VER_1).
SSTACB_TBLOFST	X'0C'	X'04'	Input	Offset from SSTACB to first control block statistics table entry.
SSTACB_NUMENT	X'10'	X'02'	Input	Number of control block statistics table entries.
SSTACB_ENTLEN	X'12'	X'02'	Input	Length of each control block statistics table entry.

The following table provides the following information about the BPE control block statistics table entry:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Each control block type in the system has one table entry.

Important: The BPE and the IMS component running on the BPE can define a block type with the same name. Use the SSTACB_F1_SYS flag to differentiate between the blocks in this case.

Table 199. BPE control block statistics table entry

Field name	Offset	Length	Field usage	Description
SSTACB_BTE	X'00'	N/A	N/A	DSECT label for BPE control block statistics table entry.
SSTACB_TYPE	X'00'	X'04'	Input	Block type.
SSTACB_FLG1	X'04'	X'01'	Input	CBTE flag 1 (unlabeled bits are reserved by IBM). **SSTACB_F1_COMP (X'20')** Blocks are compressible. **SSTACB_F1_SYS (X'10')** Block is a BPE block. **SSTACB_F1_FIXED (X'08')** Block storage is page fixed.

Table 199. BPE control block statistics table entry (continued)

Field name	Offset	Length	Field usage	Description
SSTACB_FLG2	X'05'	X'01'	Input	CBTE flag 2 (unlabeled bits are reserved by IBM). **SSTACB_F2_31ONLY (X'10')** Block is in 31-bit only storage. **SSTACB_F2_PAGE (X'02')** Get BPAGE on 4 KB page boundary. **SSTACB_F2_ANY (X'01')** Block in LOC=ANY storage.
SSTACB_IDX	X'06'	X'01'	Input	Block index number.
SSTACB_SP	X'07'	X'01'	Input	Block storage subpool.
SSTACB_#GET	X'08'	X'04'	Input	Number of gets for this block type.
SSTACB_CURBYTES	X'0C'	X'04'	Input	Current number bytes in pool.
SSTACB_MAXBYTES	X'10'	X'04'	Input	Maximum number bytes in pool.
SSTACB_#GETMAIN	X'14'	X'04'	Input	Number of GETMAINs for BPAGEs.
SSTACB_#FREEMAIN	X'18'	X'04'	Input	Number of FREEMAINs for BPAGEs.
SSTACB_CURBLKS	X'1C'	X'04'	Input	Current number blocks in pool.

The following table provides the following information about the BPE AWE services statistics area:

- The field name
- The offset
- The length
- The field usage
- A description of the field

The AWE services area contains a header with global statistics and information, followed by a table with one entry for each instance of an AWE server running in the system.

Table 200. BPE AWE services statistics area

Field name	Offset	Length	Field usage	Description
SSTAAW	X'00'	N/A	N/A	DSECT label for BPE AWE services statistics area.
SSTAAW_ID	X'00'	X'04'	Input	AWE services section eye catcher ("AWE").
SSTAAW_LENGTH	X'04'	X'04'	Input	Length of AWE section (includes AWE server statistics table).
SSTAAW_VERSION	X'08'	X'04'	Input	AWE services statistics version number. The current version is X'00000001' (SSTAAW_VER_1).
SSTAAW_TBLOFST	X'0C'	X'04'	Input	Offset from SSTAAW to first AWE statistics table entry.
SSTAAW_NUMENT	X'10'	X'02'	Input	Number of AWE server statistics table entries.

Table 200. BPE AWE services statistics area (continued)

Field name	Offset	Length	Field usage	Description
SSTAAW_ENTLEN	X'12'	X'02'	Input	Length of each AWE server statistics table entry.

The following table provides the following information about the BPE AWE services statistics table entry:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Each active AWE server running in the system has one table entry.

Important: It is possible for BPE and the IMS component running on the BPE to define an AWE server type with the same name. Use the SSTAAW_F1_SYS flag to differentiate between the identically-named BPE and user-product AWE servers.

Table 201. BPE AWE services statistics table entry

Field name	Offset	Length	Field usage	Description
SSTAAW_ASTE	X'00'	N/A	N/A	DSECT label for AWE server statistics table entry.
SSTAAW_TYPE	X'00'	X'04'	Input	AWE server type.
SSTAAW_SERVID	X'04'	X'04'	Input	Unique server ID number.
SSTAAW_FLG1	X'08'	X'01'	Input	Flag 1 (AQSB_FLG1) (unlabeled bits are reserved by IBM). **SSTAAW_F1_INIT X'80'** Server init is in progress. **SSTAAW_F1_TERM X'40'** All servers should terminate. **SSTAAW_F1_MULTI X'20'** Queue is multi-server.
SSTAAW_FLG2	X'09'	X'01'	Input	Flag 2 (AQHE_FLG1) (unlabeled bits are reserved by IBM). **SSTAAW_F2_GENERIC X'80'** Generic AWE server. **SSTAAW_F2_AUTO X'40'** Server is AUTOSTARTed. **SSTAAW_F2_SYSTCB X'20'** Server runs under system TCB. **SSTAAW_F2_SYS X'10'** System (BPE) server. **SSTAAW_F2_LOC24 X'08'** Thread blocks in 24-bit storage. **SSTAAW_F2_FORCEMAX X'04'** Force maximum threads on AUTOSTART.

Table 201. BPE AWE services statistics table entry (continued)

Field name	Offset	Length	Field usage	Description
SSTAAW_TCBID	X'0A'	X'01'	Input	ID number of owning TCB.
SSTAAW_NUMTHDS	X'0B'	X'01'	Input	Number of server threads for this queue header.
SSTAAW_QHDR	X'0C'	X'04'	Input	Address of AWE queue header.
SSTAAW_IDX	X'10'	X'01'	Input	AWE server index number (LVL001).
SSTAAW_MAXCONSECAWE	X'11'	X'01'	Input	Maximum consecutive AWEs this server will process before issuing a voluntary BPEWAIT to let other threads run (0 = no limit) (LVL001).
	X'12'	X'02'	Input	Reserved (LVL001).
SSTAAW_TQCOUNT	X'14'	X'04'	Input	Number times an extra server was woken up off of the AQSB_THREADQ (multi-server queue headers only).
SSTAAW_NUMAWE	X'18'	X'1C'	Input	Number of AWEs processed off of this queue header.
SSTAAW_NUMEQS	X'1C'	X'04'	Input	Number of times one or more AWEs were dequeued from this queue header (NUMAWE/NUMDEQS is the average number of AWEs on the queue header).
SSTAAW_PROCTIME	X'20'	X'08'	Input	Cumulative time spent in processing routine for this queue header, in STCK units.
SSTAAW_NOWORK	X'28'	X'04'	Input	Number of times an AWE server was woken up and found no work (multi-server queue headers only).
SSTAAW_READYWAIT	X'2C'	X'04'	Input	Number of times an AWE server had to wait for access to the AWE ready queue (multi-server queue headers only).
SSTAAW_MAXCAWECT	X'30'	X'04'	Input	Number of times an AWE server thread processes more consecutive AWEs than the maximum number (in field SSTAAW_MAXCONSECAWE) and issued a BPEWAIT message to voluntarily give up the DU if other ready work exists. (LVL001)
SSTAAW_IDLEWORKCT	X'34'	X'04'	Input	Number of idles on this server's AWE queue header after processing one or more AWEs. When no AWEs are processed, field SSTAAW_NOWORK counts the number of idles. (LVL001)

Table 201. BPE AWE services statistics table entry (continued)

Field name	Offset	Length	Field usage	Description
SSTAAW_MAXAWESTCK	X'38'	X'08'	Input	The hardware STCK value that corresponds to the time when the value in SSTAAW_MAXAWEPROC is reached. The field is 0 if no AWEs are processed since the last time the BPE statistics user exit is called. (LVL001)
SSTAAW_MAXAWEPROC	X'40'	X'04'	Input	The maximum number of AWEs processed between two successive idles (waits for work) of an AWE server thread during the past BPE user exit statistics interval. This field is not cumulative and is reset to 0 after the BPE statistics user exit is called so that a new maximum can be collected for the next statistics interval. (LVL001)
	X'44'	X'0C'		Reserved. (LVL001)

The following table provides the following information about the BPE storage services statistics area:

- The field name
- The offset
- The length
- The field usage
- A description of the field

Table 202. BPE storage services statistics area

Field name	Offset	Length	Field usage	Description
SSTASG	X'00'	N/A	N/A	DSECT label for BPE storage services statistics area.
SSTASG_ID	X'00'	X'04'	Input	Storage services section eye catcher ("STG").
SSTASG_LENGTH	X'04'	X'04'	Input	Length of storage section.
SSTASG_VERSION	X'08'	X'04'	Input	Storage services statistics version number. The current version is X'00000001' (SSTASG_VER_1).
	X'0C'	X'0C'	Input	Reserved.
SSTASG_STGPVT24	X'18'	X'04'	Input	Number of bytes of private storage currently allocated in 24-bit storage by the BPE GETMAIN service, BPEGETM. Note that the values for the stack, control block, and buffer pool services are included in this number.

Table 202. BPE storage services statistics area (continued)

Field name	Offset	Length	Field usage	Description
SSTASG_STGPVT31	X'1C'	X'04'	Input	Number of bytes of private storage allocated in 31-bit storage by the BPE GETMAIN service, BPEGETM. Note that the values for the stack, control block, and buffer pool services are included in this number.
SSTASG_STKPVT24	X'20'	X'04'	Input	Number of bytes of private storage currently allocated in 24-bit storage by the BPE stack manager service.
SSTASG_STKPVT31	X'24'	X'04'	Input	Number of bytes of private storage currently allocated in 31-bit storage by the BPE stack manager service.
SSTASG_CBPVT24	X'28'	X'04'	Input	Number of bytes of private storage currently allocated in 24-bit storage by the BPE control block service.
SSTASG_CBPVT31	X'2C'	X'04'	Input	Number of bytes of private storage currently allocated in 31-bit storage by the BPE control block service.
SSTASG_BPPVT24	X'30'	X'04'	Input	Number of bytes of private storage currently allocated in 24-bit storage by the BPE buffer pool service.
SSTASG_BPPVT31	X'34'	X'04'	Input	Number of bytes of private storage currently allocated in 31-bit storage by the BPE buffer pool service.

Chapter 7. BPE-based DBRC user exit routines

The BPE-based DBRC user exit routines enable you to run the existing DBRC user exit routines in a BPE (Base Primitive Environment).

Related reference

"DBRC Command Authorization exit routine (DSPDCAX0)" on page 322
The DBRC Command Authorization exit routine (DSPDCAX0) can be used to verify that a user is authorized to issue a particular command or DBRC application programming interface (API) request.

"DBRC SCI registration exit routine (DSPSCIX0)" on page 325
The DBRC SCI Registration exit routine (DSPSCIX0), formerly called the SCI Registration exit routine, supplies the IMSplex name needed for DBRC's Structured Call Interface (SCI) registration.

"RECON I/O exit routine (DSPCEXT0)" on page 418
The RECON I/O exit routine (DSPCEXT0) tracks changes to the RECON data set, which you can log in a journal. You can use the journal, in turn, as a trace facility, to monitor the activity of specific record types, or to write your own recovery utility for the RECON data set.

DBRC Request exit routine

The DBRC Request exit routine activates both before and after DBRC request processing and allows a user-supplied program to interrogate DBRC request information. Although doing so is not recommended, the exit routine can also be used to bypass standard DBRC request processing.

Subsections:

About this routine

The DBRC Request exit routine is an optional exit routine, and is a diagnosis, modification, or tuning interface.

To call this exit, set the EXITDEF statement in the BPE user exit list PROCLIB member to TYPE=REQUEST. When the EXITDEF statement is set to TYPE=REQUEST, all user exits on the list are always called. Setting the value of UXPL_CALLNEXTP to UXPL_CALLNEXTNO does not prevent user exits on the list from being called. The START request function invokes all the listed exits in the specified order. The END request function invokes all the listed exits in reverse order. All of the defined user exits are always called regardless of the setting of UXPL_CALLNEXTP, the exit function code, or the setting of BRQX_DONOTCALL.

In addition to providing an access point for user-supplied programs to interrogate DBRC request information, the DBRC request user exit also includes an option to bypass standard DBRC request processing.

Attention: The results of bypassing DBRC request processing are unpredictable. Bypassing DBRC request processing is not recommended.

Table 203. DBRC request exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL

Table 203. DBRC request exit routine attributes (continued)

Attribute	Description
Naming convention	Using standard z/OS conventions, you can give the routine any name up to 8 characters in length. Be sure that the name is unique and does not conflict with the existing members of the data set in which this routine is stored. Because most IMS-supplied routines begin with the prefixes BPE, CQS, CSL, DFS, DBF, DSP, DXR, IMS, or HWS, choose a name that does not begin with these letters.
Binding	You must bind this routine into an authorized data set as a reentrant (RENT) load module.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS callable services. It is eligible to use BPE user exit callable services.
Sample routine location	No sample routine is provided.

Communicating with IMS

IMS uses the entry and exit registers to communicate with this type of user-supplied exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the BPE user exit parameter list (mapped by macro BPEUXPL)
13	Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.
14	Return point address of the exit routine.
15	Entry point address of the exit routine.

The following table describes the DBRC Request exit routine parameters.

Table 204. DBRC Request user exit parameter list

Field name	Offset	Length in bytes	Field Usage	Description
BRQX_ID	X'00'	X'08'	Input	Eye catcher "DSPBRQX"
BRQX_LEN	X'08'	X'04'	Input	Length of the DSPBRQX block
BRQX_PVER	X'0C'	X'04'	Input	Parameter list version number
BRQX_FUNC	X'10'	X'04'	Input	Function code: **1** Start request processing (BRQX_FUNC_START) **2** End request processing (BRQX_FUNC_END)
BRQX_BRLSBPTR	X'14'	X'04'	Input	Address of DFSBRLSB
BRQX_ASCD	X'18'	X'04'	Input	Address of SCD

Table 204. DBRC Request user exit parameter list (continued)

Field name	Offset	Length in bytes	Field Usage	Description
BRQX_Flags	X'1C'	X'01'	Output	Miscellaneous flags: **X'80'** Do not call DBRC request processing routine (BRQX_DONOTCALL). If BRQX_FUNC = BRQX_FUNC_START (function code 1) and BRQX_DONOTCALL is set, DBRC request processing is bypassed. If there are multiple DBRC request user exits defined, the setting of BRQX_DONOTCALL can be altered by any exit on the list.
	X'1D'	X'0F'	None	Reserved

Contents of registers on exit

Register	Contents
15	Register 15 contains the return code. Any return code other than 0 is ignored.

Related concepts

DBRC API (System Programming APIs)

Related reference

BPE exit list members of the IMS PROCLIB data set (System Definition)

DBRC Security exit routine

The DBRC Security exit routine can be used to verify that a user is authorized to issue a particular command or DBRC application programming interface (API) request.

Subsections:

About this routine

The DBRC Security exit routine is an optional exit routine and is selected using the INIT.RECON or CHANGE.RECON commands. The exit can be used with RACF or another security product. The security product is invoked first, and return and reason codes are passed to the routine. The return code then determines the success or failure of the authorization. The exit overrides the outcome of the security product. DBRC messages issued as a result of unsuccessfully invoking the security product are suppressed.

This exit routine is required if the COMMAND AUTH setting in the RECON status record is EXIT or BOTH, and an EXITDEF statement exists in the BPE user exit list PROCLIB member for a TYPE=SECURITY. Without an EXITDEF statement, the DBRC Command Authorization exit routine is called.

If an EXITDEF statement exists, when the exit is invoked, all user exits of a TYPE=SECURITY are called in the order specified by the EXIT= keyword.

Table 205. Command authorization exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL

Table 205. Command authorization exit routine attributes (continued)

Attribute	Description
Naming convention	Using standard z/OS conventions, you can give the routine any name up to 8 characters in length. Be sure that the name is unique and does not conflict with the existing members of the data set in which this routine is stored. Because most IMS-supplied routines begin with the prefixes BPE, CQS, CSL, DFS, DBF, DSP, DXR, IMS, or HWS, choose a name that does not begin with these letters.
Binding	You must bind this routine into an authorized data set as a reentrant (RENT) load module.
Including the routine	No special steps are needed to include this routine. The exit is only included if DBRC command authorization (CMDAUTH) is set to EXIT or BOTH.
IMS callable services	This exit is not eligible to use IMS callable services. It is eligible to use BPE user exit callable services.
Sample routine location	DSPDCAX0 is provided in the IMS.SDFSSMPL data set, and you can modify it to work in both BPE and non-BPE DBRC environments.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routines.

Contents of registers on entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the BPE user exit parameter list (mapped by macro BPEUXPL)
13	Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.
14	Return address
15	Entry point address of exit routine

On entry to the DBRC Security exit routine, register 1 points to a standard BPE user exit parameter list. In that list, the field UXPL_EXITPLP contains the address of the DBRC Security user exit routine parameter lists (mapped by the DBRC command authorization (DCA) interface parameter block (DSPDCABK). The parameters are described in the following table.

The following fields are used only for the non-BPE DBRC command authorization exit routine (DSPCAX0) and are set to 0 for this routine:

- DCAExitAddr
- DCAUserAreaPtr
- DCAUserAreaLen

Table 206. DBRC Security User Exit parameter list

Field name	Offset	Length in bytes	Field Usage	Description
DCABLKID	X'00'	X'08'	Input	Eye catcher "DSPCABK"
DCABLKLN	X'08'	X'04'	Input	Length of the block

Table 206. DBRC Security User Exit parameter list (continued)

Field name	Offset	Length in bytes	Field Usage	Description
DCARNPTR	X'0C'	X'04'	Input	Address of the resource name (RN)
DCARNLEN	X'10'	X'04'	None	Resource name length
DCARHPTR	X'14'	X'04'	Input	Address of RN high-level qualifier
DCARHLEN	X'18'	X'04'	Input	Length of RN high-level qualifier
DCARVPTR	X'1C'	X'04'	Input	Address of RN command verb
DCARVLEN	X'20'	X'04'	Input	Length of RN command verb
DCARMPTR	X'24'	X'04'	Input	Address of RN command modifier
DCARMLEN	X'28'	X'04'	Input	Length of RN command modifier
DCARQPTR	X'2C'	X'04'	Input	Address of RN command qualifier
DCARQLEN	X'30'	X'04'	Input	Length of RN command qualifier
DCAUserID	X'34'	X'08'	Input	User ID of command issuer
DCAExitAddr	X'3C'	X'04'	None	Address is 0 for BPE user exit
DCAFlags	X'40'	X'04'	Input	Miscellaneous flags: **X'80'** Security product was called. **X'40'** Security exit DSPDCAX0 was called. **X'20'** 1st call (REQUEST=LIST) done. **X'10'** DBRC API Request **X'08'** BPE user exit was called
DCASAFRetCode	X'44'	X'04'	Input	Security product (RACF or equivalent) return code
DCARACFRetCode	X'48'	X'04'	Input	RACF return code
DCARACFRsnCode	X'4C'	X'04'	Input	RACF reason code
DCAExitRetCode	X'50'	X'04'	Output	Security exit return code
DCAUserAreaPtr	X'54'	X'04'	Input	Address is 0 for BPE user exit
DCAUserAreaLen	X'58'	X'04'	Input	Length is 0 for BPE user exit
DCARACRReq	X'5C'	X'08'	Input	RACROUTE request type
DCAVersion	X'64'	X'04'	Input	Parameter list version number (00000001)
	X'68'	X'20'	None	Reserved

Contents of registers on exit

Before returning to DBRC, the exit routine must restore all registers except for register 15, which contains the following return code.

Register	Contents
15	Return code: **0** User is authorized to use the DBRC command. **Non-zero** Reject the command because of an unauthorized user ID. This return code is ignored unless the exit routine is one of the following: • The exit routine is the last routine defined in the exit list for the security exit. • The exit routine sets the byte pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO.
	All other registers must be restored.

Related reference

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

"DBRC Command Authorization exit routine (DSPDCAX0)" on page 322
The DBRC Command Authorization exit routine (DSPDCAX0) can be used to verify that a user is authorized to issue a particular command or DBRC application programming interface (API) request.

Sample DBRC Security Exit Routine

Use the sample DBRC Security exit routine (DSPDCAX0) to verify that a user is authorized to issue a particular command or DBRC application programming interface (API) request.

RECON I/O exit routine

A BPE-based DBRC gives control to the RECON I/O exit routine during I/O operations to the RECON data set. This exit routine performs the function of DSPCEXT0 for BPE-based DBRCs.

Subsections:

About this routine

The RECON I/O exit routine tracks changes to the RECON data set, which you can log in a journal. You can code the RECON I/O exit routine so that it updates the journal each time a record of the data set is updated, inserted, deleted, or read. You can also record changes that are internal to the RECON access modules, such as header record extension control item changes, or the addition and deletion of multiple update control records within the data set.

You can use the journal, in turn, as a trace facility, to monitor the activity of specific record types, or as a means of writing your own recovery utility for the RECON data set.

To call the RECON I/O exit routine, the EXITDEF statement in the BPE user exit list PROCLIB member must be set for a TYPE=RECONIO. If the EXITDEF statement is not specified, the user exit DSPCEXT0 will be called. When the EXITDEF statement is specified, all user exits of TYPE=RECONIO will be called in the order specified by the EXITS=keyword.

You can use the RECON I/O exit routine when RECON access is either serial or parallel.

Recommendation: Do not use the BPE REFRESH USEREXIT for a DBRC address space during periods of high activity. No DBRC request processing will occur while the BPE user exit is being refreshed.

The following table shows the attributes of the RECON I/O exit routine.

Table 207. RECON I/O exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, and DCCTL.
Naming convention	Using standard z/OS conventions, you can give the routine any name up to 8 characters in length. Be sure that the name is unique and does not conflict with the existing members of the data set in which this routine is stored. Because most IMS-supplied routines begin with the prefixes BPE, CQS, CSL, DFS, DBF, DSP, DXR, IMS, or HWS, choose a name that does not begin with these letters.
Binding	You must write and bind this routine as reentrant (RENT).
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS callable services. It is eligible to use BPE user exit callable services.
Sample routine location	The IMS.ADFSSRC data set contains member name DSPCEXT1, which you can modify to provide support for both BPE and non-BPE based DBRC environments. DSPCEXT1 must be linked as DSPCEXT0

You must write and bind the RECON I/O exit routine as reentrant. It is entered from DBRC in 31-bit addressing mode and must return to DBRC in 31-bit addressing mode. All parameters and data areas supplied to RECON I/O exit routine by DBRC are located above the 16 MB line.

If the RECON I/O exit routine terminates abnormally, calls to the routine will continue to be made up to the limit specified by the ABLIM parameter.

Calling the routine

Control is passed to the RECON I/O exit routine whenever a RECON record has been successfully read, written, or modified on COPY 1 of the RECON data set, not necessarily for every physical I/O operation. Changes to the header record extension also cause the RECON I/O exit routine to be called.

When RECON access is parallel, the RECON data set can be accessed by multiple DBRC instances concurrently. In this case, multiple instances of the RECON I/O exit routine can be invoked concurrently.

With serial access, the user can rely on all updates written to the RECON data set. If an error occurs, and the update is backed out by DBRC, the exit is called for all the updates made during backout. If the exit is used to mirror updates, the exit can immediately make the equivalent updates to a mirror data set.

With parallel access, the backout of data is not done by DBRC, which means that the exit is not called for the backout updates. Updates made during a given series should not be considered hardened in the RECON data set until a commit call is made. If the exit is used to mirror updates, it must either be capable of backing out the updates it mirrors, or it must collect all updates for a given series and only mirror them if the exit is called with a commit call.

Whenever a record of data is inserted, updated, deleted, or read, this routine is called after the call or change is made to the RECON data set. For each insert, delete, and read call, the routine receives a copy of the inserted, deleted, or read record, respectively. For each update call, the routine receives a copy of the record as it appeared both before and after it was updated. For delete and update calls, the copy of the record read must be incomplete if DBRC is unable to locate all segments for that record. In this case, byte 2 of word 17 in the I/O exit parameter list is set to X'40'.

The records passed to the exit routine are in the format of the release level of the RECON data set, and rather than the release level of the DBRC that calls the exit. In order for the DBRCs of multiple IMS systems at different release levels to coexist, the RECON data set must be at the level of the highest level system. An indication of the RECON data set release level exists in the parameter list that is passed to the exit. When the RECON is upgraded to a new release, the exit routine can use both the old release format

and the new release format. During the upgrade process, the release level in the parameter list shows the old release level. A flag in the parameter list indicates that an upgrade is in progress.

The release level of the RECON can change from one Begin Series call to another. Except during the upgrade process, the release level does not change between the Begin Series call and the Terminate Series call.

Any modifications to storage that this routine makes must be made to storage that is obtained by the routine, not to the data areas pointed to by DBRC or IMS or to those contained within the routine itself.

Each series of I/O accesses that DBRC makes to the RECON data set is indicated to the routine by a Begin Series call. When the series of I/O operations is complete, the routine receives a Terminate Series call.

Performance recommendations

While this routine is running, the RECON data set is reserved so that no other jobs can access RECON records. To minimize the affect that this routine's execution has on your system's performance, you need to:

- Limit the I/O operations that the routine itself performs and simplify the routine's functions to make efficient use of processing time.
- Be sure that any resources needed solely by the routine (that is, those not needed by DBRC/IMS in general) are immediately available to z/OS when DBRC is initialized and in control. You should therefore avoid operations that can put the routine, and therefore DBRC, in a prolonged wait state (for example, the ENQUEUE/DEQUEUE of resources that cannot be readily accessed by the routine or write to operator messages that require waiting for a reply).
- Be aware that with parallel RECON access, the RECON data set is not reserved. In addition, multiple instances of the RECON I/O exit routine can be invoked concurrently.

DBRC enables the size of a record in the RECON data set to not be limited by the defined RecordSize. DBRC divides its own records into segments, each of which fits into a single Control Interval (CI) and is sent by VSAM as a complete record. Segmenting allows a logical RECON record to be as large as 16 777 215 bytes. The RECON I/O exit routine will be presented with complete, unsegmented logical records.

To minimize the performance impact that the routine's execution has on DBRC, the routine spools its copy of RECON data records to a data set (specified by a DD statement with the name DBRCDATA) for later offline processing outside the DBRC environment. Any data sets that your routine references need to be accessed by DD statements as well.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the BPE user exit parameter list (mapped by macro BPEUXPL)
13	Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.
14	Return address
15	Entry point address of exit routine

Description of parameters

When control is given to the BPE-based DBRC RECON I/O exit routine, register 1 contains the address of the standard BPE user exit parameter list (BPEUXPL). In this list, the field UXPL_EXITPLP contains he address of the RECON I/O user exit parameter list, which is mapped by the DBRC RECON I/O interface parameter block (DSPPRIOX).

This routine receives the parameter list from the calling RECON access module at the first Begin Series call for a job. The parameter list points to the same data area for all subsequent calls for that job.

The data area pointed to by the parameter list is 24 words (96 bytes) long and starts on a fullword boundary. Fields 9 through 16 of the list are free to be used by the exit routine and remain unchanged by DBRC after the first Begin Series call. They initially contain all zeros.

The first byte of word 17 of the list indicates the release level of the RECON in hexadecimal format. RECON release levels by IMS version are:

Version	RECON release level in hexadecimal format
IMS Version 13	X'D1'
IMS Version 12	X'C1'
IMS Version 11	X'B1'

Byte 2 of Field 17 contains flags. Bytes 3 and 4 of Field 17, and Fields 22 through 24 are reserved for future use.

The following tables list the exit parameter list at various exit points in the routine.

Table 208. Begin Series parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 1 - "Begin Series" A binary **1**, indicating a Begin Series call to this routine as a result of a RESERVE function having been performed on the RECON data set.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. • All calls to the exit for this series, including the Terminate Series call, will have this request token. • For serial RECON access, the token is the RESERVE sequence number from the control record extension. This number is incremented by one in the control record extension each time DBRC completes a successful RESERVE of the RECON data set. • For parallel RECON access, the token is a store-clock (STCK) value captured before the RECON I/O exit is invoked.

Table 208. Begin Series parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_CHANGECNT	X'10'	X'04'	Input	Changed record count. • For serial RECON access, this is the changed record count from the control record extension. The changed record count is a 32-bit logical value that can eventually wrap back to zero. This is the count as of the last DEQUEUE function that DBRC performs, or that value plus one if the last DBRC abended. A change to the RECON data set has occurred if an ENQUEUE sequence detects that the last DBRC abended. For more information about the changed record count, see the "Terminate Series" exit call in this topic. • For parallel RECON access, the count is always zero. The RECON I/O exit routine interprets a zero count to mean that parallel access is in effect.
	X'14'	X'2C'	None	Reserved.
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. For IMS Version 12, the value is X'C1'. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	• For serial RECON access, binary zeroes. • For parallel RECON access, the DBRC instance token. This token is a binary value that can be used to distinguish the calls in a given series in case two DBRC instances present the same STCK value (request token). The instance token is unique across currently executing DBRC instances.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 209. Insert record parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 3 - "Insert". A binary 3, indicating an insert call to this routine as a result of a record having been inserted into the RECON data set.
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
RIOX_RECORDLEN	X'10'	X'04'	Input	Length of the record that has been inserted.
RIOX_RECORDADR	X'14'	X'04'	Input	Address of the record that has been inserted.
	X'18'	X'28'	None	Reserved.

Table 209. Insert record parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 210. Update record parameter list

Field Name	Offset	Length	Field Usage	Meaning or content
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	A binary **4**, indicating an update call to this routine as a result of a record having been updated on the RECON data set.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. Unchanged from the Begin Series call.
RIOX_OLDRECLEN	X'10'	X'04'	Input	Length of the record image before update.
RIOX_OLDRECADR	X'14'	X'04'	Input	Address of a copy of the record as it appeared before the update.
RIOX_NEWRECLEN	X'18'	X'04'	Input	Length of the replacement record.
RIOX_NEWRECADR	X'1C'	X'04'	Input	Address of the replacement record.
	X'20'	X'20'	None	Reserved.
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is ON if the record, before being changed, had a missing segment. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 211. Delete record parameter list

Field Name	Offset	Length	Field Usage	Meaning or content
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 5 - "Delete". A binary 5, indicating a delete call to this routine as a result of a record having been deleted from the RECON data set.

Table 211. Delete record parameter list (continued)

Field Name	Offset	Length	Field Usage	Meaning or content
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
RIOX_OLDRECLEN	X'10'	X'04'	Input	Length of the record that has been deleted.
RIOX_OLDRECADR	X'14'	X'04'	Input	Address of the record that has been deleted.
	X'18'	X'28'	None	Reserved.
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 212. Read record parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	Function Code 6 - "Read". A binary 6, indicating a read call to this routine as a result of a record having been read from the RECON data set.
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
RIOX_OLDRECLEN	X'10'	X'04'	Input	Length of the record that has been read.
RIOX_OLDRECADR	X'14'	X'04'	Input	Address of the record that's been read.
	X'18'	X'28'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 213. Commit request parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"

Table 213. Commit request parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_FUNC	X'04'	X'04'	Input	Function Code 7 - "Commit". A binary 7, indicating a commit call to this routine. The call results from previous updates (including inserts and deletes) for this current series being committed to the RECON data set. This call is made only for parallel RECON access.
RIOX_TOKEN	X'08'	X'08'	None	Request token. Unchanged from the Begin Series call.
	X'10'	X'30'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains flags that are defined as follows: • Bit 0 is ON when upgrade is in progress • Bit 1 is OFF • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 214. Backout request parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"
RIOX_FUNC	X'04'	X'04'	Input	A binary **8**, indicating a backout call to this routine. The call results from previous updates (including inserts and deletes) for this current series being backed out of the RECON data set. This call is made only for parallel RECON access.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. Unchanged from the Begin Series call.
	X'10'	X'30'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	Unchanged from the Begin Series call.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Table 215. Terminate Series parameter list

Field Name	Offset	Length	Field Usage	Description
RIOX_EYEC	X'00'	X'04'	Input	Eye catcher "CEXT"

Table 215. Terminate Series parameter list (continued)

Field Name	Offset	Length	Field Usage	Description
RIOX_FUNC	X'04'	X'04'	Input	Function Code 2 - "Terminate Series" A binary **2**, indicating a Terminate Series call to this routine. This call occurs at the end of processing a DBRC request. For serial RECON access, the DEQUEUE for the RECON has been performed.
RIOX_TOKEN	X'08'	X'08'	Input	Request token. Unchanged from the Begin Series call.
RIOX_CHANGECNT	X'10'	X'04'	Input	Final changed record count. • For serial RECON access, this is the final changed record count as it now appears on the control record extension. The changed record count is a 32-bit logical value that can eventually wrap back to zero. Either the count is the same as the Begin Series call value, or it is that value plus one if any change has been made (other than to the record extension itself) to the RECON data set since the Begin Series call. By monitoring the value of this counter between its value here and the next Begin Series exit call, you can detect changes made to the RECON data set by other occurrences of DBRC. • For parallel RECON access, the count is always zero. With parallel access, you cannot detect when changes have been made to the RECON by other DBRC instances.
	X'14'	X'2C'	None	Reserved
RIOX_FLAGS	X'40'	X'04'	Input	Byte 1 indicates the release level of the RECON. Byte 2 contains the following flags: • Bit 0 is ON when upgrade is in progress. • Bit 1 is OFF. • Bit 2 is ON for parallel RECON access. Bytes 3 and 4 are reserved.
RIOX_INSTANCE_TOKEN	X'44'	X'10'	Input	• For serial RECON access, binary zeroes. • For parallel RECON access, the DBRC instance token. This token is a binary value that can be used to distinguish the calls in a given series in case two DBRC instances present the same STCK value (request token). The instance token is unique across currently executing DBRC instances.
RIOXVersion	X'54'	X'04'	Input	Parameter list version number (00000001)
	X'58'	X'08'	None	Reserved.

Contents of registers on exit

Before returning to DBRC, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return code	Meaning
0	Always zero

Related concepts

Initializing and maintaining the RECON data sets (System Administration)

Related reference

"RECON I/O exit routine (DSPCEXT0)" on page 418
The RECON I/O exit routine (DSPCEXT0) tracks changes to the RECON data set, which you can log in a journal. You can use the journal, in turn, as a trace facility, to monitor the activity of specific record types, or to write your own recovery utility for the RECON data set.

"Routine binding restrictions" on page 9
If you bind DL/I exit routines, you must keep in mind the recommendations and restrictions in this information.

Sample RECON I/O exit routine

Use the sample RECON I/O exit routine to confirm that DBRC gives control to the RECON I/O exit routine during I/O operations to the RECON.

To minimize the performance impact that the routine's execution has on DBRC, the routine spools its copy of RECON data records to a data set (specified by a DD statement with the name DBRCDATA) for later off-line processing outside the DBRC/IMS environment. Any data sets that your routine references need to be accessed by DD statements as well.

DBRC statistics

You can use the BPE Statistics user-supplied exit to gather both BPE and DBRC statistics.

When the BPE Statistics user-supplied exit routine is run, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user-supplied exit parameter list, BPESTXP, contains the pointer to the DBRC statistics header.

Subsections:

- "DBRC statistics header" on page 543
- "DBRC statistics record DSPBST1" on page 543
- "DBRC statistics record DSPBST2" on page 545

DBRC statistics header

The following table describes the contents of the DBRC Statistics header. The statistics header is mapped by DSPBSTX.

Table 216. DBRC statistics header data

Field Name	Offset	Length	Field usage	Description
BSTX_ID	X'00'	X'08'	Input	Eye catcher "DSPBSTX".
BSTX_LEN	X'08'	X'04'	Input	Length of header.
BSTX_PVER	X'0C'	X'04'	Input	Header version number (X'00000001').
BSTX_CLIENT_CNT	X'10'	X'04'	Input	Number of active clients for which statistics are available.
BSTX_ST1OFF	X'14'	X'04'	Input	Offset to statistics area for first client. The offset points to the DSPBST1 area.
	X'18'	X'04'	None	Reserved.

DBRC statistics record DSPBST1

Record DSPBST1 contains statistics that are related to VSAM I/O, RESERVE information, and other overall performance related information for a DBRC client. The statistics are cumulative. Some of the data in record DSPBST1 might reflect information for a request that is being processed when the BPE statistics exit routine is called.

The following table describes the DBRC statistics record DSPBST1.

Table 217. DBRC statistics record DSPBST1

Field name	Offset	Length	Field usage	Description
BST1_ID	X'00'	X'08'	Input	Eye catcher "DSPBST1".
BST1_LEN	X'08'	X'04'	Input	Length of valid data.
BST1_BST2OFF	X'0C'	X'04'	Input	Offset of request data for this client. The offset is from the beginning of the DSPBST1 block.
BST1_NEXTCLIENT	X'10'	X'04'	Input	Offset to next client's DSPBST1 data. The offset is from the beginning of the current DSPBST1 block.
BST1_PVER	X'14'	X'04'	Input	Parameter list version number (X'00000001').
	X'18'	X'08'	None	Reserved.
BST1_CLIENTID	X'20'	X'10'	Input	ID of DBRC client (IMS SYSID).
	X'30'	X'10'	None	Reserved.
BST1_LOCATE	X'40'	X'04'	Input	Number of DBRC requests to LOCATE a RECON record.
BST1_CHANGE	X'44'	X'04'	Input	Number of DBRC requests to CHANGE a RECON record.
BST1_WRITE	X'48'	X'04'	Input	Number of DBRC requests to WRITE a RECON record.
BST1_DELETE	X'4C'	X'04'	Input	Number of DBRC requests to DELETE a RECON record.
BST1_GET	X'50'	X'04'	Input	Number of VSAM GET requests made by DBRC.
BST1_PUT	X'54'	X'04'	Input	Number of VSAM PUT requests made by DBRC.
BST1_ERASE	X'58'	X'04'	Input	Number of VSAM ERASE requests made by DBRC.
	X'5C'	X'04'	None	Reserved.
BST1_RSVCNT	X'60'	X'04'	Input	Number of RESERVE requests.
	X'64'	X'04'	None	Reserved.
BST1_RSVWAIT	X'68'	X'08'	Input	Accumulated time waiting for a reserve.
BST1_RSVALLN	X'70'	X'04'	Input	Number of requests to RESERVE all RECON data sets.
	X'74'	X'04'	None	Reserved.
BST1_RSVALLW	X'78'	X'08'	Input	Accumulated time waiting for a reserve of ALL data sets.
BST1_RQSTCNT	X'80'	X'04'	Input	Number of original requests to DBRC.
	X'84'	X'04'	None	Reserved.
BST1_RQSTTIME	X'88'	X'08'	Input	Accumulated time spent processing DBRC requests.
BST1_RETRYCNT	X'90'	X'04'	Input	Number of times a request required being retried.
	X'94'	X'04'	None	Reserved.
BST1_RETRYTIM	X'98'	X'08'	Input	Accumulated amount of time spent on requests that required retry. This does not include the time spent processing the request the last time (when it no longer required a retry).
BST1_PREEMPTN	X'A0'	X'04'	Input	Number of group services requests that preempted the processing of another DBRC request. In other words, they were processed after a DBRC request was received, but before returning to the request caller.
	X'A4'	X'04'	None	Reserved.
BST1_PREEMPTIME	X'A8'	X'08'	Input	Accumulated time spent processing preemptive requests.
BST1_DDLKCNT	X'B0'	X'04'	Input	Number of VSAM deadlocks encountered.
BST1_TIMEOUT	X'B4'	X'04'	Input	Number of VSAM timeouts encountered.
BST1_REOPEN	X'B8'	X'04'	Input	Number of VSAM errors that required closure and reopening of the RECON data sets.

Table 217. DBRC statistics record DSPBST1 (continued)

Field name	Offset	Length	Field usage	Description
	X'BC'	X'04'	None	Reserved.
BST1_RETRYOP	X'C0'	X'04'	Input	Number of VSAM OPEN requests that were retried.
BST1_CMITERR	X'C4'	X'04'	Input	Number of z/OS Resource Recovery Services commit failures.
BST1_BACKERR	X'C8'	X'04'	Input	Number of RRS backout failures.
	X'CC'	X'04'	None	Reserved.
BST1_BATCH_ENQ_CNT	X'D0'	X'04'	Input	Number of batch enqueues on DSPURI02 qname.
	X'D4'	X'04'	None	Reserved.
BST1_BATCH_ENQ_TIME	X'D8'	X'08'	Input	Accumulated time spent waiting for enqueue on DSPURI02 qname.
BST1_QUEUETIME	X'E0'	X'08'	Input	Average time request waits to be processed by DBRC.
BST1_MAXQUEUETIME	X'E8'	X'08'	Input	Maximum time a request waited to be processed by DBRC.
BST1_MAX_REQUEST_CNT	X'F0'	X'04'	Input	For parallel request processing, the maximum number of requests being processed in parallel. For serial processing, the maximum number of requests waiting to be processed.
	X'F4'	X'0C'	None	Reserved.

DBRC statistics record DSPBST2

Record DSPBST2 contains statistics that are related to specific DBRC request types. The statistics are cumulative. Some of the data in record DSPBST1 might reflect information for a request that is being processed when the BPE statistics exit routine is called.

The following table describes the DBRC statistics record DSPBST2.

Table 218. DBRC statistics record DSPBST2

Field name	Offset	Length	Field usage	Description
BST2_ID	X'00'	X'08'	Input	Eye-catcher "DSPBST2".
BST2_LEN	X'08'	X'04'	Input	Length of valid data.
BST2_PVER	X'0C'	X'04'	Input	Parameter list version number (X'00000001').
BST2_rqstlist	X'10'	X'04'	Input	Offset to first entry of the request data information mapped by DSPBST_RQ. Offset is from the beginning of the DSPBST2 block.
BST2_rqstcount	X'14'	X'02'	Input	Number of request data information entries.
BST2_rqstLen	X'16'	X'02'	Input	Length of a request data information entry.
	X'18'	X'08'	None	Reserved.

The following table describes the DBRC request entry statistics record BST2_REQUEST_DATA, which represents an entry in the DSPBST2 statistics record. The address of the first entry is the address of the DSPBST2 record plus the offset value in BST2_rqstlist. To obtain the address of subsequent entries, add BST2_rqstLen to the address of the current entry.

Table 219. DBRC request entry statistics record BST2_REQUEST_DATA

Field name	Offset	Length	Field usage	Description
BST2_rqst_ID	X'00'	X'01'	Input	Request ID - this is the BRLBF2 value in the DFSBRLSB control block used for the request.
	X'01'	X'03'	None	Reserved.
BST2_rqst_cnt	X'04'	X'04'	Input	Number of requests for this function.
BST2_rqst_retry	X'08'	X'04'	Input	Number of times that the function required a retry.
	X'0C'	X'04'	None	Reserved.
BST2_rqst_time	X'10'	X'08'	Input	Total time spent processing this function.
	X'18'	X'08'	None	Reserved.
BST2_rqst_loc	X'20'	X'04'	Input	Number of record LOCATE requests made processing this function.
BST2_rqst_chg	X'24'	X'04'	Input	Number of record CHANGE requests made processing this function.
BST2_rqst_del	X'28'	X'04'	Input	Number of record DELETE requests made processing this function.
BST2_rqst_wrt	X'2C'	X'04'	Input	Number of record WRITE requests made processing this function.

Related reference

“BPE Statistics user-supplied exit routine” on page 511
The BPE Statistics user-supplied exit routine is called at regular intervals during the life of a BPE address space, and a final time at normal address shutdown, to gather address-space related statistics.

Chapter 8. BPE-based CQS user-supplied exit routines

Use BPE-based CQS user exit routines to customize and monitor your CQS environment.

Note: Throughout this topic the term "user exit routine" means "user-supplied exit routine."

This topic contains Product-sensitive Programming Interface information.

You write these exit routines, no samples are provided. The CQS user exit routines receive control in the CQS address space in an authorized state. CQS uses Base Primitive Environment (BPE) services to call and manage the CQS user exit routines.

A list of the user exit routines and their functions follows:

CQS Initialization-Termination
Called during CQS initialization and CQS normal termination.

CQS Client Connection
Called when a client connects to or disconnects from a structure.

CQS Queue Overflow
Called during overflow processing to verify queue name eligibility for overflow processing.

CQS Structure Statistics
Called at the end of CQS system checkpoint to allow structure-related statistics gathering.

CQS Structure Event
Called during processing for structure processing-related event notification.

In addition, you can use the BPE Statistics User exit to gather CQS statistics.

General user-supplied exit routine interface information for CQS

CQS uses BPE services to call and manage its user exit routines. BPE allows you to externally specify the user exit routine modules to be called for a particular exit routine type using EXITDEF= statements in BPE user exit PROCLIB members. BPE also provides a common user exit routine execution environment. This environment includes:

- Standard BPE user exit parameter list
- Static work areas for the routines
- Dynamic work areas for the routines
- Callable services for the routines
- A recovery environment to protect against abends in the user exit routines

Recommendation: Write CQS user exit routines in assembler, not in a high level language. CQS does not support exit routines running under Language Environment for z/OS. If you write an exit routine in a high level language, and that routine is executing in the Language Environment for z/OS, you might have abends or performance problems. Language Environment for z/OS is designed for applications running in key 8, problem program state. CQS user exit routines execute in key 7 supervisor state.

Related reading

- For complete information about displaying and refreshing user exits, see *IMS Version 15.2 Commands, Volume 1: IMS Commands A-M*.

Related reference

“CQS client exit routines” on page 625

CQS client exit routines allow a CQS client to monitor the CQS environment.

CQS initialization-termination user-supplied exit routine

The Initialization-Termination (Init-Term) exit routine is called during CQS initialization and CQS normal termination. The Init-Term exit routine is not called during CQS abnormal termination.

This exit routine is optional.

The CQS Init-Term user exit routine is driven for the following events:

- CQS initialization, after CQS has completed its initial processing, but before it connects to any structures.
- CQS normal termination, during CQS address space termination, after CQS has disconnected from all structures.

The Init-Term exit routine is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit PROCLIB member. You can specify one or more user exit routines of this type. When this exit routine is invoked, the exit routines are driven in the order they are specified by the EXITS= keyword.

Recommendation: Write the Init-Term exit routine so that it is reentrant. It is invoked AMODE 31.

Contents of registers on entry

Register
Contents

1
Address of the"Standard BPE user exit parameter list" on page 485. The exit routine-specific parameter list pointed to by the UXPL_EXITPLP field is mapped by macro CQSINTMX.

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Always set this to zero.

All other registers must be restored.

CQS initialization and termination parameter lists

On entry to the Init-Term exit routine register 1 points to a Standard BPE user exit parameter list. The field UXPL_EXITPLP in this list contains the address of the Init-Term user exit routine parameter lists (mapped by the CQSINTMX macro). The parameters are described in the following two tables.

Table 220. CQS init-term user-supplied exit routine parameter list: CQS initialization

Field name	Offset	Length	Field Usage	Description
ITXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001')
ITXFUNC	X'04'	X'04'	Input	Function code **1** CQS initialization (ITXFINIT)
ITXCQSID	X'08'	X'08'	Input	CQS identifier
ITXCQSVN	X'10'	X'04'	Input	CQS version number

Table 221. CQS init-term user-supplied exit routine parameter list: CQS termination

Field name	Offset	Length	Field usage	Description
ITXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001')
ITXFUNC	X'04'	X'04'	Input	Function code **2** CQS normal termination (ITXFNTRM)
ITXCQSID	X'08'	X'08'	Input	CQS identifier
ITXCQSVN	X'10'	X'04'	Input	CQS version number

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CQS client connection user-supplied exit routine

This exit routine is called when a client connects to or disconnects from a structure.

This exit routine is optional.

The Client Connection exit routine is driven for the following events:

- Client connect; after a client successfully connects to one or more structures.
- Client disconnect; after a client disconnects normally or abnormally from one or more structures.

The Client Connection exit routine is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit PROCLIB member. You can specify one or more user exit routines of this type. When this exit routine is invoked, all user exit routines of this type are driven in the order specified by the EXITS= keyword.

Recommendation: Write the Client Connection exit routine so that it is reentrant. It is invoked AMODE 31.

Contents of registers on entry

Register
Contents

1
Address of the “Standard BPE user exit parameter list” on page 485. The UXPL_EXITPLP field contains the address of the CQS Client Connection exit parameter list, which is mapped by macro CQSCLNCX.

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Always set this to zero.

All other registers must be restored.

CQS client connection and disconnect parameter lists

On entry to the Client Connection exit routine, R1 points to a Standard BPE user exit parameter list. The field UXPL_EXITPLP in this list contains the address of the Client Connection user exit routine parameter list (mapped by the CQSCLNCX macro). The parameters for client connection and client disconnect are described in the following two tables.

Table 222. CQS client connection user-supplied exit routine parameter list: client connection

Field name	Offset	Length	Field usage	Description
CCXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
CCXFUNC	X'04'	X'04'	Input	Function code **1** Client connect (CCXFCONN).
CCXCQSID	X'08'	X'08'	Input	CQS identifier.
CCXCQSVN	X'10'	X'04'	Input	CQS version number.
CCXCLNNM	X'14'	X'08'	Input	Client name.
CCXCSNUM	X'1C'	X'04'	Input	Number of structure name entries in the list.
CCXCSENL	X'20'	X'04'	Input	Length of each structure name list entry.
CCXCSLST	X'24'	X'04'	Input	Address of first structure name entry. Each entry contains the 16-byte name of a structure that the client connected to.

Table 223. CQS client connection user-supplied exit routine parameter list: client disconnect

Field name	Offset	Length	Field usage	Description
CCXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
CCXFUNC	X'04'	X'04'	Input	Function code **2** Client disconnect (CCXFDISC).
CCXCQSID	X'08'	X'08'	Input	CQS identifier.
CCXCQSVN	X'10'	X'04'	Input	CQS version number.
CCXCLNNM	X'14'	X'08'	Input	Client name.
CCXDFLG1	X'1C'	X'01'	Input	Flag byte indicates whether the client disconnect is abnormal **X'80'** Client disconnect is abnormal (CCXDABND).
N/A	X'1D'	X'03'		Reserved.
CCXDSNUM	X'20'	X'04'	Input	Number of structure name entries in the list.
CCXDSENL	X'24'	X'04'	Input	Length of each structure name list entry.
CCXDSLST	X'28'	X'04'	Input	Address of first structure name entry. Each entry contains the 16-byte name of a structure that the client disconnected from.

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CQS Queue overflow user-supplied exit routine

The Queue Overflow exit routine is called during overflow queue selection processing to approve or veto a queue name for overflow processing.

This exit routine is optional.

During overflow processing the Queue Overflow exit routine is called to verify that a queue name selected by CQS is eligible for overflow processing. When CQS determines that the structure has reached its overflow threshold, overflow threshold processing begins. Then CQS determines which queues are using the most storage in the structure. The queues using the most storage in the structure become candidates for overflow and are moved to the overflow structure. Or, if no overflow structure is defined, the queues using the most storage in the structure no longer allow CQSPUT requests for the queue.

Restriction: The queue overflow user exit does not apply to the resource structure.

During queue selection processing the Queue Overflow exit routine is invoked once per selected queue name to approve or veto the queue name for overflow processing. If the exit routine approves the move or the exit routine is not specified, all data objects for that queue (such as IMS messages for that destination) are moved to the overflow structure. All additional processing for that queue name is done in the overflow structure, if the overflow structure exists. If no overflow structure exists, CQSPUT requests to the queue are rejected. If the move is vetoed, the queue name is removed from the overflow candidate list, and another queue name is selected.

The Queue Overflow exit routine is defined as TYPE=OVERFLOW in the EXITDEF statement in the BPE user exit PROCLIB member. You can specify one or more user exit routines of this type. When this exit routine is invoked, all such routines are driven in the order specified by the EXITS= keyword.

Because multiple overflow exit routines might exist, the last exit routine called is the one that determines whether the queue name is selected for overflow. If an exit routine accepts a queue name as one that is valid for overflow processing or does not recognize the name, the exit routine must set R15 to 0 and specify that the next exit in the list should be called. This allows the next exit routine to have a chance to veto the name selection. If an exit routine determines that a queue name is ineligible as a candidate for overflow processing, the exit routine must set R15 to 4 and specify that no more exit routines are to be called.

Within the Standard BPE user exit parameter list is the field UXPL_CALLNEXTP, which is a pointer to a byte of storage which is set by the exit routine to indicate whether the next exit routine in the list is to be called. When the byte of storage is set to UXPL_CALLNEXTYES, the next exit is called (if one exists). When the byte of storage is set to UXPL_CALLNEXTNO, no more exits are called for this queue name.

If a Queue Overflow exit routine determines that a queue name is not a candidate for overflow, the exit routine can set the byte pointed to by field UXPL_CALLNEXTP to the value of UXPL_CALLNEXTNO (X'04') so that no other exit routines are called for the queue name.

Recommendation: Write the Queue Overflow exit routine so that it is reentrant. It is invoked AMODE 31.

Contents of registers on entry

Register
Contents

1
Address of the“Standard BPE user exit parameter list” on page 485. The exit routine-specific parameter list pointed to by the UXPL_EXITPLP field is mapped by macro CQSQOFLX.

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Allow queue to be moved to overflow structure.

4
Do not move queue to overflow structure; select another candidate.

Attention: This return code is ignored unless the exit routine is the last overflow user exit called for the queue name.

An exit routine is considered the last one called when either of the following are true:

1. The exit routine is the last routine defined in the exit list for the overflow queue.
2. The exit routine sets the byte pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO.

All other registers must be restored.

CQS queue overflow parameter list

On entry to the Queue Overflow exit routine, R1 points to a Standard BPE user exit parameter list. The field UXPL_EXITPLP in this list contains the address of the CQS Queue Overflow user exit routine parameter list (mapped by the CQSQOFLX macro). The parameters are described in detail in the following table.

Table 224. CQS queue overflow user-supplied exit routine parameter list

Field name	Offset	Length	Field usage	Description
QOXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
QOXFUNC	X'04'	X'04'	Input	Function code **1** Queue name selection (QOXFQOFL).
QOXQOFL1	X'08'	X'01'	Input	Flag byte indicating whether this is the first overflow exit call for this overflow threshold process. The exit routine is called once per selected queue name for each occurrence of overflow threshold processing. This bit will be on for the first queue name for an occurrence of overflow threshold processing. **X'80'** This is the initial entry for this overflow threshold process (QOXQ11ST)
N/A	X'09'	X'03'	None	Reserved.
QOXCQSID	X'0C'	X'08'	Input	CQS identifier.
QOXCQSVN	X'14'	X'04'	Input	CQS version number.
QOXSTRNM	X'18'	X'10'	Input	Structure name.
QOXQNAME	X'28'	X'10'	Input	Queue name selected for overflow processing.
QOXDOBJN	X'38'	X'04'	Input	Number of data objects on the selected queue name.

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CQS structure statistics user-supplied exit routine

The CQS Structure Statistics user exit routine enables you to gather statistics related to the structure.

This exit routine is optional.

The exit routine is driven at the end of a successful system checkpoint. All statistical data that CQS gathers, including rebuild statistics and checkpoint statistics, are passed to the Structure Statistics user exit at the end of each successful system checkpoint. All statistical data is logged in the Structure Statistics log record. You can also obtain this same statistical data with the CQSQUERY FUNC=STRSTAT request.

Recommendation: Some statistics about resource structures are passed in the structure statistics. CQS system checkpoint does not apply to resource structures. Use the STATINV parameter in the BPE configuration PROCLIB member to define the time interval so that BPE regularly drives CQS's statistics

user exit. See *IMS Version 15.2 System Definition* for more information about the BPE configuration PROCLIB member.

The CQS Structure Statistics user exit routine is defined as TYPE = STRSTAT in the EXITDEF statement in the BPE user exit PROCLIB member. You can specify one or more user exit routines of this type. When this exit routine is invoked, all routines of this type are driven in the order specified by the EXITS= keyword.

Recommendation: Write the CQS Structure Statistics exit routine so that it is reentrant. It is invoked AMODE 31.

Subsections:

Contents of registers on entry

Register
Contents

1
Address of the standard "Standard BPE user exit parameter list" on page 485. The UXPL_EXITPLP field in this list contains the address of the CQS Structure Statistics exit parameter list, which is mapped by macro CQSSTATX.

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Always set this to zero.

All other registers must be restored.

CQS structure statistics user-supplied exit routine parameter list

On entry to the Structure Statistics exit routine, R1 points to a Standard BPE user exit parameter list. The field UXPL_EXITPLP in this list contains the address of the CQS Structure Statistics user exit routine parameter list (mapped by the CQSSTATX macro). The parameters are described in the following table.

Table 225. CQS structure statistics user-supplied exit routine parameter list

Field name	Offset	Length	Field usage	Description
SAXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
SAXFUNC	X'04'	X'04'	Input	Function code **1** System checkpoint (SAXFCSYS).
SAXCQSID	X'08'	X'08'	Input	CQS identifier.
SAXCQSVN	X'10'	X'04'	Input	CQS version number.
SAXSTRNM	X'14'	X'10'	Input	Structure name.
SAXSSTT1	X'24'	X'04'	Input	Address of structure process statistics record for activity performed by CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT1 macro). See the following section for a description of the process statistics record.
SAXSSTT2	X'28'	X'04'	Input	Address of CQS request statistics record for activity performed for CQS requests for this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT2 macro).
SAXSSTT3	X'2C'	X'04'	Input	Address of data object statistics record for activity performed on data objects in this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT3 macro). See Table 228 on page 557 for a description of the object statistics record.
SAXSSTT4	X'30'	X'04'	Input	Address of queue name statistics record for activity performed on queue names in this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT4 macro). See Table 229 on page 559 for a description of the queue name statistics record.
SAXSSTT5	X'34'	X'04'	Input	Address of z/OS request statistics record for activity performed by CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT5 macro). See Table 230 on page 559 for a description of the z/OS request statistics record.
SAXSSTT6	X'38'	X'04'	Input	Address of rebuild statistics record containing data from the last rebuild in which this CQS acted as master (mapped by the CQSSSTT6 macro). See Table 231 on page 560 for a description of the rebuild statistics record.
SAXSSTT7	X'3C'	X'04'	Input	Address of structure checkpoint statistics record containing data from the last three structure checkpoints in which this CQS acted as master (mapped by the CQSSSTT7 macro). See Table 232 on page 564 for a description of the structure checkpoint statistics record.

CQS structure process statistics record

The following table describes the CQS Structure Statistics user exit routine structure process statistics record.

Table 226. CQS structure process statistics record

Field name	Offset	Length	Field usage	Description
SS1ID	X'00'	X'08'	Input	Eye catcher CQSSSTT1
SS1LN	X'08'	X'04'	Input	Length of valid data
SS1PVSN	X'0C'	X'04'	Input	Parameter list version number (X'00000002')
SS1YCHKP	X'10'	X'04'	Input	Number of times CQS successfully performed system checkpoint processing for the structure
SS1TCHKP	X'14'	X'04'	Input	Number of times CQS successfully performed structure checkpoint processing for the structure
SS1RBLD	X'18'	X'04'	Input	Number of times CQS successfully performed rebuild processing for the structure
SS1OFLWT	X'1C'	X'04'	Input	Number of times CQS performed overflow threshold processing for the structure
SS1DUPLX	X'20'	X'04'	Input	Number of times CQS successfully established a duplexing rebuild

CQS request statistics record

The following table describes the Structure Statistics user exit routine CQS request statistics record.

Table 227. CQS request statistics record

Field name	Offset	Length	Field usage	Description
SS2ID	X'00'	X'08'	Input	Eye catcher CQSSSTT2
SS2LN	X'08'	X'04'	Input	Length of valid data
SS2PVSN	X'0C'	X'04'	Input	Parameter list version number (X'00000002')
SS2BRWSE	X'10'	X'04'	Input	Number of CQSBRWSE requests for this structure
SS2CHKPT	X'14'	X'04'	Input	Number of CQSCHKPT requests for this structure
SS2CONN	X'18'	X'04'	Input	Number of CQSCONN requests for this structure
SS2DEL	X'1C'	X'04'	Input	Number of CQSDEL requests for this structure
SS2DISC	X'20'	X'04'	Input	Number of CQSDISC requests for this structure
SS2INFRM	X'24'	X'04'	Input	Number of CQSINFRM requests for this structure
SS2MOVE	X'28'	X'04'	Input	Number of CQSMOVE requests for this structure
SS2PUT	X'2C'	X'04'	Input	Number of CQSPUT requests for this structure
SS2QUERY	X'30'	X'04'	Input	Number of CQSQUERY requests for this structure
SS2READ	X'34'	X'04'	Input	Number of CQSREAD requests for this structure
SS2RECVR	X'38'	X'04'	Input	Number of CQSRECVR requests for this structure
SS2RSYNC	X'3C'	X'04'	Input	Number of CQSRSYNC requests for this structure
SS2UNLCK	X'40'	X'04'	Input	Number of CQSUNLCK requests for this structure
SS2UPD	X'44'	X'04'	Input	Number of CQSUPD requests for this structure

Data object statistics record for CQS

The following table describes the Structure Statistics user exit routine data object statistics record.

Table 228. Data object statistics record

Field name	Offset	Length	Field usage	Description
SS3ID	X'00'	X'08'	Input	Eye catcher CQSSSTT3.
SS3LN	X'08'	X'04'	Input	Length of valid data.
SS3PVSN	X'0C'	X'04'	Input	Parameter list version number (X'00000003').
SS3PTOBJ	X'10'	X'04'	Input	Number of data objects added to the structure with COMMIT = NO. This count does not include data objects added with COMMIT = YES or RECOVERABLE = NO.
SS3PTCMT	X'14'	X'04'	Input	Number of data objects added to the structure with COMMIT = YES. This count indicates the number of recoverable UOWs added to the structure. This count plus the number of data objects that are added with COMMIT = NO is the total number of recoverable data objects added to the structure.
SS3PTNRO	X'18'	X'04'	Input	Number of data objects added to the structure with RECOVERABLE = NO. This count indicates the number of nonrecoverable UOWs added to the structure. This count plus the number of data objects that are added with COMMIT = YES is the total number of UOWs that were added to the structure.
SS3RDOBJ	X'1C'	X'04'	Input	Number of data objects read from the structure.
SS3MVOBJ	X'20'	X'04'	Input	Number of data objects moved from one queue to another on the structure.
SS3ULOBJ	X'24'	X'04'	Input	Number of data objects unlocked on the structure.
SS3CROBJ	X'28'	X'04'	Input	Number of data objects created.
SS3UPOBJ	X'2C'	X'04'	Input	Number of data objects updated.
SS3ENTAL	X'30'	X'04'	Input	Number of data entries allocated on the primary structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full.
SS3ENTIN	X'34'	X'04'	Input	Number of data entries in use on the primary structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full.
SS3ENTHI	X'38'	X'04'	Input	High water mark for number of data entries on the primary structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full.
SS3ENTTM	X'3C'	X'08'	Input	Time stamp representing the time the data entry high water mark was reached for the primary structure (in STCK format).
SS3ELMAL	X'44'	X'04'	Input	Number of data elements allocated on the primary structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full.

Table 228. Data object statistics record (continued)

Field name	Offset	Length	Field usage	Description
SS3ELMIN	X'48'	X'04'	Input	Number of data elements in use on the primary structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full.
SS3ELMHI	X'4C'	X'04'	Input	High water mark for number of data elements on the primary structure. Compare the data element high water mark field to the data element allocated field to determine the closest the structure came to becoming full.
SS3ELMTM	X'50'	X'08'	Input	Time stamp representing the time the data element high water mark was reached for the primary structure (in STCK format).
Reserved	X'58'	X'14'	Input	
SS3OENAL	X'6C'	X'04'	Input	Number of data entries allocated on the overflow structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full. This field is present only if SS3PVSN is 3 or greater.
SS3OENIN	X'70'	X'04'	Input	Number of data entries in use on the overflow structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full. This field is present only if SS3PVSN is 3 or greater.
SS3OENHI	X'74'	X'04'	Input	High water mark for number of data entries on the overflow structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full. This field is present only if SS3PVSN is 3 or greater.
SS3OENTM	X'78'	X'08'	Input	Time stamp representing the time the data entry high water mark was reached for the overflow structure (in STCK format). This field is present only if SS3PVSN is 3 or greater.
SS3OELAL	X'80'	X'04'	Input	Number of data elements allocated on the overflow structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full.
SS3OELIN	X'84'	X'04'	Input	Number of data elements in use on the overflow structure. Compare the data entry in use field to the data entry allocated field to determine how close the structure is to becoming full. This field is present only if SS3PVSN is 3 or greater.
SS3OELHI	X'88'	X'04'	Input	High water mark for number of data elements on the overflow structure. Compare the data element high water mark field to the data element allocated field to determine the closest the structure came to becoming full. This field is present only if SS3PVSN is 3 or greater.

Table 228. Data object statistics record (continued)

Field name	Offset	Length	Field usage	Description
SS3OELTM	X'8C'	X'08'	Input	Time stamp representing the time the data element high water mark was reached for the overflow structure (in STCK format). This field is present only if SS3PVSN is 3 or greater.
Reserved	X'94'	X'14'	Input	

Queue name statistics record for CQS

The following table describes the Structure Statistics user exit routine queue name statistics record.

Restriction: The queue name statistics record does not apply to resource structures.

Table 229. Queue name statistics record

Field name	Offset	Length	Field usage	Description
SS4ID	X'00'	X'08'	Input	Eye catcher CQSSSTT4
SS4LN	X'08'	X'04'	Input	Length of valid data
SS4PVSN	X'0C'	X'04'	Input	Parameter list version number (X'00000001')
SS4INFQN	X'10'	X'04'	Input	Number of queue names for which an inform was performed
SS4UNFQN	X'14'	X'04'	Input	Number of queue names for which an uninform was performed
SS4NFYQN	X'18'	X'04'	Input	Number of queue name notifications (when a queue goes from empty to non-empty)

z/OS request statistics record for CQS

The following table describes the Structure Statistics user exit routine z/OS request statistics record.

Table 230. z/OS request statistics record

Field name	Offset	Length	Field usage	Description
SS5ID	X'00'	X'08'	Input	Eye catcher CQSSSTT5.
SS5LN	X'08'	X'04'	Input	Length of valid data.
SS5PVSN	X'0C'	X'04'	Input	Parameter list version number (X'00000002').
SS5IXGWR	X'10'	X'04'	Input	Number of IXGWRITE requests for the structure. This represents the number of log records written during processing on the structure.
SS5IXGBR	X'14'	X'04'	Input	Number of IXGBRWSE requests for the structure.
SS5IXLDQ	X'18'	X'04'	Input	Number of IXLLIST DEQ_EVENTQ requests for the structure.
SS5IXLWR	X'1C'	X'04'	Input	Number of IXLLIST WRITE requests for the structure.
SS5IXLRD	X'20'	X'04'	Input	Number of IXLLIST READ requests for the structure.
SS5IXLMV	X'24'	X'04'	Input	Number of IXLLIST MOVE requests for the structure.
SS5IXLDL	X'28'	X'04'	Input	Number of IXLLIST DELETE requests for the structure.

Table 230. z/OS request statistics record (continued)

Field name	Offset	Length	Field usage	Description
SS5IXLMG	X'2C'	X'04'	Input	Number of IXLMG requests for the structure.
SS5IXLUS	X'30'	X'04'	Input	Number of IXLUSYNC requests for the structure.
SS5IXEWR	X'34'	X'04'	Input	Number of IXLLSTE WRITE requests for the structure.
SS5IXERD	X'38'	X'04'	Input	Number of IXLLSTE READ requests for the structure.
SS5IXMRL	X'3C'	X'04'	Input	Number of IXLLSTM READ_LIST requests for the structure.
SS5IXEDL	X'40'	X'04'	Input	Number of IXLLSTE DELETE requests for the structure.
SS5IXMDL	X'44'	X'04'	Input	Number of IXLLSTM DELETE_ENTRYLIST requests for the structure.

Structure rebuild statistics record for CQS

Structure rebuild statistics are gathered only by the CQS that is the master of the structure rebuild process. A CQS has access only to the data it gathers. Each CQS keeps structure rebuild statistics for the last rebuild for which it was the master.

The following table describes the Structure Statistics user exit routine structure rebuild statistics record.

Table 231. Structure rebuild statistics record

Field name	Offset	Length	Field usage	Description
SS6ID	X'00'	X'08'	Input	Eye catcher CQSSSTT6.
SS6LN	X'08'	X'04'	Input	Length of valid data.
SS6PVSN	X'0C'	X'04'	Input	Parameter list version number (X'00000003').
SS6ELMIO	X'10'	X'04'	Input	Data elements in use on old structure.
SS6ELMAO	X'14'	X'04'	Input	Data elements allocated on old structure.
SS6ENTIO	X'18'	X'04'	Input	Data entries in use on old structure (data object count).
SS6ENTAO	X'1C'	X'04'	Input	Data entries allocated on old structure.
SS6MCIO	X'20'	X'04'	Input	Event monitoring controls (EMCs) in use on old structure (active informs).
SS6EMCAO	X'24'	X'04'	Input	EMCs in use on old structure (active informs).
SS6SIZEO	X'28'	X'04'	Input	Old structure size in 4 KB blocks.
SS6CFTO	X'2C'	X'04'	Input	Old CF total space in 4 KB blocks.
SS6CFFO	X'30'	X'04'	Input	Old CF free space in 4 KB blocks.
SS6CFNMO	X'34'	X'08'	Input	Old CF name in which structure was allocated before rebuild.
	X'3C'	X'04'		Unused.
SS6ELMIN	X'40'	X'04'	Input	Data elements in use on new structure.
SS6ELMAN	X'44'	X'04'	Input	Data elements allocated on new structure.
SS6ENTIN	X'48'	X'04'	Input	Data entries in use on new structure (data object count).
SS6ENTAN	X'4C'	X'04'	Input	Data entries allocated on new structure.

Table 231. Structure rebuild statistics record (continued)

Field name	Offset	Length	Field usage	Description
SS6EMCIN	X'50'	X'04'	Input	EMCs in use on new structure (active informs).
SS6EMCAN	X'54'	X'04'	Input	EMCs in use on new structure (active informs).
SS6SIZEN	X'58'	X'04'	Input	New structure size in 4 KB blocks.
SS6CFTN	X'5C'	X'04'	Input	New CF total space in 4 KB blocks.
SS6CFFN	X'60'	X'04'	Input	New CF free space in 4 KB blocks.
SS6CFNMN	X'64'	X'08'	Input	New CF name in which structure is allocated after rebuild.
	X'6C'	X'04'		Unused.
SS6RBTIM	X'70'	X'08'	Input	Rebuild time stamp (STCK).
SS6POPCT	X'78'	X'04'	Input	Repopulation from SRDS count (RCVRY) or objects copied count (COPY).
SS6MVQCT	X'7C'	X'04'	Input	Entries moved to moveq during phase 2 count.
SS6PUTCT	X'80'	X'04'	Input	Entries written during phase 3 count.
SS6MOVCT	X'84'	X'04'	Input	Entries moved during phase 3 count.
SS6OBJCT	X'88'	X'04'	Input	Data objects affected by recovery count (recoverable and nonrecoverable).
SS6UOWCT	X'8C'	X'04'	Input	UOWs affected by recovery count (recoverable and nonrecoverable).
SS6FLAG1	X'90'	X'01'	Input	Flag byte. **X'40'** Duplexing is established. **X'80'** These statistics are for the last rebuild performed for the structure.
SS6FLAG2	X'91'	X'01'	Input	Rebuild flag. Indicates the last rebuild or duplexing rebuild event received that updated these rebuild statistics: **1** Structure rebuild statistics. **2** Duplexing started statistics. **3** Duplexing ended statistics and z/OS switched to simplex structure (either old or new structure).
	X'92'	X'02'		Unused.
The remaining fields of this table apply to rebuild failures. The CQS0242E message identifies the rebuild failure reason.				
The following fields apply to rebuild failures that occurred while rebuild was processing a CQS log record. Use this information to locate the log record in the CQS log to give to an IBM service representative.				

Table 231. Structure rebuild statistics record (continued)

Field name	Offset	Length	Field usage	Description
SS6LGTYP	X'94'	X'01'	Input	Log record type of log record being processed when rebuild failure occurred.
SS6LGSUB	X'95'	X'01'	Input	Log record subtype of log record being processed when rebuild failure occurred.
SS6STYPE	X'96'	X'01'	Input	Structure type of log record being processed when rebuild failure occurred.
	X'97'	X'01'		Unused.
SS6LGTIM	X'98'	X'08'	Input	Log record time stamp of log record being processed when rebuild failure occurred.
SS6CQSID	X'A0'	X'08'	Input	CQS ID associated with log record being processed when rebuild failure occurred.
SS6CLNTN	X'A8'	X'08'	Input	Client name associated with log record being processed when rebuild failure occurred.
SS6SRCQ	X'B0'	X'10'	Input	Source client or private queue name associated with log record being processed when rebuild failure occurred.
SS6DSTQ	X'C0'	X'10'	Input	Destination queue name associated with log record being processed when rebuild failure occurred.
SS6UOW	X'D0'	X'20'	Input	UOW associated with log record being processed when rebuild failure occurred.
SS6UNIQ1	X'F0'	X'04'	Input	Information unique to log record or rebuild data object entry when rebuild failure occurred.
SS6UNIQ2	X'F4'	X'04'	Input	Information unique to log record or rebuild data object entry when rebuild failure occurred.
SS6UNIQ3	X'F8'	X'04'	Input	Information unique to log record or rebuild data object entry when rebuild failure occurred.
The following fields apply to rebuild failures that occurred while rebuild was processing an IXL request to access the structure.				
SS6IXLMC	X'FC'	X'01'	Input	IXL macro that failed and caused rebuild to fail. See CQSTRACE macro for IXL macro type.
SS6IXLRQ	X'FD'	X'01'	Input	IXL request that failed and caused the rebuild to fail.
	X'FE'	X'02'		Unused.
SS6IXLRC	X'100'	X'04'	Input	IXL return code returned by IXL request that caused rebuild to fail.
SS6IXLRN	X'104'	X'04'	Input	IXL reason code returned by IXL request that caused rebuild to fail.
SS6SRVRC	X'108'	X'04'	Input	This field applies to rebuild failures that occurred while rebuild was processing a service (for example, CQSTBL, BPELAGET, BPECBGET). It provides the return code of the service that failed.
	X'10C'	X'04'		Unused.

Table 231. Structure rebuild statistics record (continued)

Field name	Offset	Length	Field usage	Description
SS6VRSNO	X'110'	X'08'	Input	Old structure version (rebuild) or primary structure version (duplexing rebuild).
SS6VRSNN	X'118'	X'08'	Input	New structure version (rebuild) or secondary structure version (duplexing rebuild).
SS6CFLVO	X'120'	X'04'	Input	Old structure CF level (rebuild) or primary structure CF level (duplexing rebuild). For a primary structure CF level, this can be a composite CF level, which is at least as high as a CF level as that which has been previously reported back to any CQS as the primary structure CF level.
SS6CFLVN	X'124'	X'04'	Input	New structure CF level (rebuild) or secondary structure CF level (duplexing rebuild). For a secondary structure CF level, this can be a composite CF level, which is at least as high as a CF level as that which has been previously reported back to any CQS as the primary structure CF level.
SS6CFNMS	X'128'	X'04'	Input	CF name in which simplex structure is located (z/OS switched to simplex structure).
SS6VALFL	X'12C'	X'02'	Input	Validity flags (EEPLSSCVALIDITYFLAGS).
	X'12E'	X'02'	Input	Not used
SS6DUPST	X'130'	X'08'	Input	Last duplexing rebuild start time (STCK). The last duplexing rebuild for this structure was initiated at this time.
SS6DUPET	X'138'	X'08'	Input	Last duplexing rebuild end time (STCK). The last duplexing rebuild stopped for this structure occurred at this time.
SS6UNAVT	X'140'	X'08'	Input	Last structure temporarily unavailable time (STCK). The structure becomes temporarily unavailable because a system-managed rebuild has been initiated, a duplexing rebuild has been initiated, or a duplexing rebuild has stopped.
SS6AVT	X'148'	X'08'	Input	Last structure available time (STCK). The structure last became available at this time, after initiation of a system-managed rebuild, initiation of a duplexing rebuild, or stopping of a duplexing rebuild.
	X'150'	X'38'	Input	Unused

Structure checkpoint statistics record for CQS

Structure checkpoint statistics are gathered only by the CQS that is the master of the structure checkpoint process. A CQS has access only to the data it gathers. Each CQS keeps structure checkpoint statistics for the last three checkpoints for which it was the master. Structure checkpoint data is not reset at the end of a structure checkpoint.

The following table describes the Structure Statistics user exit routine structure checkpoint statistics record.

Table 232. Structure checkpoint statistics record

Field name	Offset	Length	Field usage	Description
SS7ID	X'00'	X'08'	Input	Eye catcher CQSSSTT7.
SS7LN	X'08'	X'04'	Input	Length of valid data.
SS7PVSN	X'0C'	X'04'	Input	Parameter list version number.
SS7FLAG1	X'10'	X'01'	Input	Flag byte. **X'80'** These statistics are from last attempted structure checkpoint taken for the structure. **X'40'** Structure Checkpoint is in progress.
	X'11'	X'03'		Unused.
SS7ENCNT	X'14'	X'04'	Input	Number of structure checkpoint statistics entries in record.
SS7ENLEN	X'18'	X'04'	Input	Length of structure checkpoint statistics entry
SS7CUR	X'1C'	X'04'	Input	Offset to current structure checkpoint statistics entry.
SS7STATS	X'20'	X'04'	Input	Start of structure checkpoint statistics entries. See the next table for a description of the structure checkpoint statistics entry.

Structure checkpoint statistics gathered by CQS

Structure checkpoint statistics are gathered only by the CQS that is the master of the structure checkpoint process. A CQS has access only to the data it gathers. Each CQS keeps structure checkpoint statistics for the last three checkpoints for which it was the master. Structure checkpoint data is not reset at the end of a structure checkpoint.

The following table describes the Structure Statistics user exit routine structure checkpoint statistics entry.

Table 233. Structure checkpoint statistics entry

Field name	Offset	Length	Field usage	Description
SS7RETCD	X'00'	X'04'	Input	Return code for this structure checkpoint
SS7QSCB	X'04'	X'08'	Input	Structure quiesce start time in STCK format
SS7QSCE	X'0C'	X'08'	Input	Structure quiesce complete time in STCK format
SS7DSPB	X'14'	X'08'	Input	Start data space/data set capture time in STCK format
SS7DSPE	X'1C'	X'08'	Input	End data space capture time in STCK format
SS7RSMB	X'24'	X'08'	Input	Structure resume start time in STCK format
SS7DSE	X'2C'	X'08'	Input	End data set capture time in STCK format
SS7CHKE	X'34'	X'08'	Input	Time when all system checkpoints completed in STCK format
SS7PELA	X'3C'	X'04'	Input	Number of allocated elements on primary structure
SS7PELU	X'40'	X'04'	Input	Number of elements in use on primary structure
SS7OELA	X'44'	X'04'	Input	Number of allocated elements on overflow structure

Table 233. Structure checkpoint statistics entry (continued)

Field name	Offset	Length	Field usage	Description
SS7OELU	X'48'	X'04'	Input	Number of elements in use on overflow structure
SS7PLEA	X'4C'	X'04'	Input	Number of allocated list entries on primary structure
SS7PLEU	X'50'	X'04'	Input	Number of list entries in use on primary structure
SS7OLEA	X'54'	X'04'	Input	Number of allocated list entries on overflow structure
SS7OLEU	X'58'	X'04'	Input	Number of list entries in use on overflow structure
SS7WRTS	X'5C'	X'04'	Input	Number of SRDS writes required

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CQS structure event user-supplied exit routine

The CQS Structure Event user exit routine is called during CQS processing to notify you of an event related to structure processing.

For certain events, CQS structure event user-supplied exit routine also allows you to gather statistics related to the structure. This exit routine is optional.

The Structure Event user exit routine applies to both resource and queue structures, but not all events are applicable to resource structures. The CQS Structure Event exit routine is driven for the following events:

- Structure Connection
  - When structure connect occurs, after CQS connects to a structure, but before rebuild or restart is performed for the structure.
  - At structure disconnect; after CQS disconnects from a structure.
- Checkpoint
  - When a system checkpoint begin, end, or failure occurs.
  - When a structure checkpoint begin, end, or failure occurs.

  Restriction: The Checkpoint event does not apply to resource structures.
- Structure Rebuild
  - When a structure copy (rebuild) begin, end, or failure occurs.
  - When a structure recovery (rebuild) begin, end, or failure occurs.

  Important: The structure failure event for a resource structure means that the structure has failed and a new structure could not be reallocated. No structure recovery is done, because resource structures do not support structure recovery.
- Structure Overflow
  - When one or more queues moved to the overflow structure.
  - When one or more queues moved from the overflow structure back to the primary structure. This event also indicates when the structure is no longer in overflow mode.

  Restriction: The Structure Overflow event does not apply to resource structures.
- Structure Status Change
  - When the structure is available again after a loss.

  - When the structure fails.
  - When CQS loses its connection to the structure.
  - When a resource structure fails and is able to allocate a new resource structure.
  - When the log stream becomes available, making the structure available.
- Structure Repopulation
  - When the structure fails and CQS is able to allocate a new resource structure.

  The Structure Repopulation event does not apply to queue structures. The client can repopulate the new resource structure with the resource data.

The exit routine is defined as TYPE=STREVENT in the EXITDEF statement in the BPE user exit PROCLIB member. You can specify one or more exit routines of this type. When this exit routine is invoked, all routines of this type are driven in the order specified by the EXITS= keyword.

Recommendation: Write the CQS Structure Event exit routine so that it is reentrant. It is invoked AMODE 31.

Subsections:

Contents of registers on entry

Register
Contents

1
Address of the "Standard BPE user exit parameter list" on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the CQS Structure Event exit parameter list, which is mapped by macro CQSSTREX.

13
Address of two pre-chained save areas. The first save area can be used by the exit routine to save registers on entry. The second save area can be used by routines that are called from the user exit routine.

14
Return address.

15
Entry point of the exit routine.

Contents of registers on exit

Register
Contents

15
Return code

0
Always set this to zero.

All other registers must be restored.

Routine parameter lists

On entry to the Structure Event exit routine, register 1 points to a Standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the CQS Structure Event user exit routine parameter list (mapped by the CQSSTREX macro).

CQS structure event exit routine parameter list

The following table describes the Structure Event user exit routine connect parameter list.

Table 234. CQS structure event user-supplied exit routine parameter list: connect

Field name	Offset	Length	Field usage	Description
STXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
STXEVENT	X'04'	X'04'	Input	Function code **1** Connect event (STXCONDS).
STXSCODE	X'08'	X'04'	Input	Event subcode **1** Structure connect (STXCONN). **2** Structure disconnect (STXDISC).
STXCQSID	X'0C'	X'08'	Input	CQS identifier.
STXCQSVN	X'14'	X'04'	Input	CQS version number.
STXSTRNM	X'18'	X'10'	Input	Structure name.
STXSTRVN	X'28'	X'08'	Input	Structure version number (mapped by the CQSSTREX macro).
STXDSTT1	X'34'	X'04'	Input	Address of structure process statistics record for activity performed by CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT1 macro). For structure disconnect only.
STXDSTT2	X'38'	X'04'	Input	Address of CQS request statistics record for activity performed for CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT2 macro). For structure disconnect only.
STXDSTT3	X'3C'	X'04'	Input	Address of data object statistics record for activity performed on data objects in this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT3 macro). For structure disconnect only.
STXDSTT4	X'40'	X'04'	Input	Address of queue name statistics record for activity performed on queue names in this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT4 macro). For structure disconnect only.

Table 234. CQS structure event user-supplied exit routine parameter list: connect (continued)

Field name	Offset	Length	Field usage	Description
STXDSTT5	X'44'	X'04'	Input	Address of z/OS request statistics record for activity performed by CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT5 macro). For structure disconnect only.
STXDSTT6	X'48'	X'04'	Input	Address of rebuild statistics record containing data from the last rebuild in which this CQS acted as master (mapped by the CQSSSTT6 macro). For structure disconnect only.
STXDSTT7	X'4C'	X'04'	Input	Address of structure checkpoint statistics record containing data from the last three structure checkpoints in which this CQS acted as master (mapped by the CQSSSTT7 macro).For structure disconnect only.

CQS structure event exit routine checkpoint parameter list

The following table describes the Structure Event user exit routine checkpoint parameter list.

Table 235. CQS structure event user-supplied exit routine parameter list: checkpoint

Field name	Offset	Length	Field usage	Description
STXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
STXEVENT	X'04'	X'04'	Input	Structure event code **2** Checkpoint event (STXCHKPT).
STXSCODE	X'08'	X'04'	Input	Structure event subcode **1** Structure checkpoint begin (STXCSTRB). **2** Structure checkpoint end (STXCSTRE). **3** Structure checkpoint failure (STXCSTRF). **4** System checkpoint begin (STXCSYSB). **5** System checkpoint end (STXCSYSE). **6** System checkpoint failure (STXCSYSF).
STXCQSID	X'0C'	X'08'	Input	CQS identifier.
STXCQSVN	X'14'	X'04'	Input	CQS version number.
STXSTRNM	X'18'	X'10'	Input	Structure Name.
STXCMCQS	X'28'	X'08'	Input	CQS identifier of the master CQS performing the checkpoint process. For system checkpoint, this is the same as the CQS identifier.

Table 235. CQS structure event user-supplied exit routine parameter list: checkpoint (continued)

Field name	Offset	Length	Field usage	Description
STXCFLG1	X'30'	X'01'	Input	Flag byte **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same (STXC1MST).
N/A	X'31'	X'03'	Input	Reserved.
STXCSTT1	X'34'	X'04'	Input	Address of structure process statistics record for activity performed by CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT1 macro). For system checkpoint end and structure checkpoint end only.
STXCSTT2	X'38'	X'04'	Input	Address of CQS request statistics record for activity performed for CQS requests on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT2 macro). For system checkpoint end and structure checkpoint end only.
STXCSTT3	X'3C'	X'04'	Input	Address of data object statistics record for activity performed on data objects in this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT3 macro). For system checkpoint end and structure checkpoint end only.
STXCSTT4	X'40'	X'04'	Input	Address of queue name statistics record for activity performed on queue names in this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT4 macro). For system checkpoint end and structure checkpoint end only.
STXCSTT5	X'44'	X'04'	Input	Address of z/OS request statistics record for activity performed by CQS processes on this structure for all clients since restart or the last successful structure checkpoint (mapped by the CQSSSTT5 macro). For system checkpoint end and structure checkpoint end only.
STXCSTT6	X'48'	X'04'	Input	Address of rebuild statistics record containing data from the last rebuild in which this CQS acted as master (mapped by the CQSSSTT6 macro). For system checkpoint end and structure checkpoint end only.
STXCSTT7	X'4C'	X'04'	Input	Address of structure checkpoint statistics record containing data from the last three structure checkpoints in which this CQS acted as master (mapped by the CQSSSTT7 macro). For system checkpoint end and structure checkpoint end only.

CQS structure event exit routine rebuild parameter list

The following table describes the Structure Event user exit routine rebuild parameter list.

Table 236. CQS structure event user-supplied exit routine parameter list: rebuild

Field name	Offset	Length	Field usage	Description
STXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
STXEVENT	X'04'	X'04'	Input	Structure event code **3** Structure rebuild event (STXRBLD).
STXSCODE	X'08'	X'04'	Input	Structure eventSubcode **1** Structure rebuild begin (STXRBLB). **2** Structure rebuild (copy) end (STXCPYE). **3** Structure rebuild (copy) failure (STXCPYF). **4** Structure rebuild failure (STXRBLF). **5** Structure rebuild (recovery) end (STXRCOVE). **6** Structure rebuild (recovery) failure (STXRCOVF).
STXCQSID	X'0C'	X'08'	Input	CQS identifier.
STXCQSVN	X'14'	X'04'	Input	CQS version number.
STXSTRNM	X'18'	X'10'	Input	Structure Name.
STXRMCQS	X'28'	X'08'	Input	CQS identifier of the master CQS performing the rebuild process.
STXRFLG1	X'30'	X'01'	Input	Flag byte **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same (STXR1MST).
N/A	X'31'	X'03'	Input	Reserved.

CQS structure event exit routine overflow parameter list

The following table describes the Structure Event user exit routine overflow parameter list.

Table 237. CQS structure event user-supplied exit routine parameter list: overflow

Field name	Offset	Length	Field usage	Description
STXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000001').
STXEVENT	X'04'	X'04'	Input	Structure event code **4** Structure overflow event (STXOVFLW).

Table 237. CQS structure event user-supplied exit routine parameter list: overflow (continued)

Field name	Offset	Length	Field usage	Description
STXSCODE	X'08'	X'04'	Input	Structure event subcode. **1** Move queues to overflow. One or more queues were selected as candidates to be moved to the overflow structure and were approved by the Queue Overflow user exit routine (STXTOOFL). **2** Move queues from overflow. One or more queues moved from the overflow structure back to the primary structure, because the queues were drained on the overflow structure. New work for these queues is placed on the primary structure (STXFROFL).
STXCQSID	X'0C'	X'08'	Input	CQS identifier.
STXCQSVN	X'14'	X'04'	Input	CQS version number.
STXSTRNM	X'18'	X'10'	Input	Structure Name.
STXOMCQS	X'28'	X'08'	Input	CQS identifier of the master CQS performing the overflow process.
STXOFLG1	X'30'	X'01'	Input	Flag byte **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same (STX01MST). **X'40'** The structure is no longer in overflow mode. This applies only to subcode 2 (STX01END).
N/A	X'31'	X'03'	Input	Reserved.
STXOLSTN	X'34'	X'04'	Input	Number of Queue Names entries in the list.
STXOLSTE	X'38'	X'04'	Input	Length of each Queue Name list entry.
STXOLSTA	X'3C'	X'04'	Input	Address of Queue Name list. Each Queue Name list entry contains the 16-byte name of a queue that is being moved to or from the overflow structure.

CQS structure event exit routine status change parameter list

The following table describes the Structure Event user exit routine status change parameter list.

Table 238. CQS structure event user-supplied exit routine parameter list: status change

Field name	Offset	Length	Field usage	Description
STXPVSN	X'00'	X'04'	Input	Parameter list version number (X'00000003').
STXEVENT	X'04'	X'04'	Input	Structure event code **5** Structure status change event (STXSCHNG).

Table 238. CQS structure event user-supplied exit routine parameter list: status change (continued)

Field name	Offset	Length	Field usage	Description
STXSCODE	X'08'	X'04'	Input	Structure event subcode **1** Structure available again after a loss (STXAVAIL). **2** The structure failed (STXFAIL). **3** CQS lost its connection to the structure (STXLCONN). **4** The log stream is becoming available, making the structure available (STXAVLOG). **Important:** This subcode applies only to queue structures. **5** The log stream is becoming available, making the structure available (STXFLOG). **Important:** This subcode applies only to queue structures. **6** The structure failed. It needs to be repopulated because this structure does not support structure recovery (STXREPOP). **Important:** This subcode applies only to resource structures.
STXCQSID	X'0C'	X'08'	Input	CQS identifier.
STXCQSVN	X'14'	X'04'	Input	CQS version number.
STXSTRNM	X'18'	X'10'	Input	Structure Name.
STXSTYPE	X'28'	X'01'	Input	Input structure type (X'01' queue structure, X'02' resource structures).
STXRSTVN	X'40'	X'08'	Input	Input structure version.

Related reference

"CQS structure statistics user-supplied exit routine" on page 553
The CQS Structure Statistics user exit routine enables you to gather statistics related to the structure.

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CQS statistics available through the BPE statistics user-supplied exit

You can use the BPE Statistics user exit to gather both BPE and CQS statistics.

When the BPE Statistics user exit is driven, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user-supplied exit parameter list, BPESTXP, contains the pointer to the CQS statistics header.

CQS statistics header

The following table describes the contents of the CQS Statistics header. The statistics header is mapped by CQSSSTTX.

Table 239. CQS statistics header data

Offset	Length	Field usage	Description
X'00'	X'08'	Input	Eye catcher "CQSSTTX"
X'08'	X'04'	Input	Length of header
X'0C'	X'04'	Input	Header version number (X'00000001')
X'10'	X'04'	Input	Number of structures for which statistics are available
X'14'	X'04'	Input	Number of statistics areas available for each structure
X'18'	X'04'	Input	Length of all statistics areas for each structure
X'1C'	X'04'	Input	Offset to statistics area for first structure (offset from CQSSSTTX)
X'20'	X'04'	Input	CQSSSTAT offset within the statistics area for each structure
X'24'	X'04'	Input	CQSSSTTI offset within the statistics area for each structure
X'28'	X'04'	Input	CQSSSTT2 offset within the statistics area for each structure
X'2C'	X'04'	Input	CQSSSTT3 offset within the statistics area for each structure
X'30'	X'04'	Input	CQSSSTT4 offset within the statistics area for each structure
X'34'	X'04'	Input	CQSSSTT5 offset within the statistics area for each structure
X'38'	X'04'	Input	CQSSSTT6 offset within the statistics area for each structure
X'3C'	X'04'	Input	CQSSSTT7 offset within the statistics area for each structure
X'40'	X'04'	Input	Reserved
X'44'	X'04'	Input	Reserved
X'48'	X'04'	Input	Reserved
X'4C'	X'04'	Input	Reserved

Related reference

“BPE user-supplied exit routine interfaces and services” on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

Chapter 9. Common Service Layer exit routines

Common Service Layer exit routines customize and monitor the ODBM, OM, RM, and SCI environments.

CSL ODBM user exit routines

You can write CSL Open Database Manager (ODBM) user exits to customize and monitor the ODBM environment. No sample exits are provided.

ODBM uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends.

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485

BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CSL ODBM Initialization and Termination user exit

Because the CSL ODBM Initialization and Termination user exit routine can be called during both Open Database Manager (ODBM) address space initialization and termination and IMSplex initialization and normal termination, you can use this exit routine, for example, to determine when these events occurred.

This exit is optional.

The CSL ODBM Initialization and Termination user exit is driven for the following events:

- ODBM initialization, after ODBM has completed initialization
- IMSplex initialization, after each IMSplex has initialized
- ODBM normal termination, when ODBM is terminating
- IMSplex normal termination, when an IMSplex is terminating

The CSL ODBM Initialization and Termination user exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are run in the order specified by the EXITS= keyword. Refer to the ODBM User Exit List PROCLIB Member in *IMS Version 15.2 System Definition* for more information on how to define user exit module names.

The CSL ODBM Initialization and Termination user exit is invoked in 31-bit addressing mode (AMODE 31) and should be reentrant.

Contents of registers on entry

On entry, the CSL ODBM Initialization and Termination user exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of 2 pre-chained save areas. The first save area may be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

Parameter list

On entry to the CSL ODBM Initialization and Termination user exit routine, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the addresses of the ODBM Initialization and Termination user exit parameter list (mapped by macro CSLDITX). Field UXPL_COMPTYPEP in this list points to the character string "ODBM" indicating an ODBM type address space.

The following table lists the user exit parameter list for ODBM Initialization and Termination user exit parameter list: ODBM initialization.

Table 240. Initialization and Termination user exit routine parameter list: ODBM initialization

Offset	Length	Field Usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **1** ODBM Initialization

The following table lists the user exit parameter list for ODBM Initialization and Termination user exit parameter list: ODBM termination.

Table 241. Initialization and Termination user exit parameter list: ODBM termination

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **2** ODBM Normal Termination

The following table lists the user exit parameter list for the CSL ODBM Initialization and Termination user exit parameter list: IMSplex initialization.

Table 242. Initialization and Termination user exit parameter list: IMSplex initialization

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **3** IMSplex Initialization
X'08'	X'08'	Input	IMSplex name

The following table lists the user exit parameter list for ODBM Initialization and Termination user exit parameter list: IMSplex termination.

Table 243. Initialization and Termination user exit parameter list: IMSplex termination

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **4** IMSplex Termination

Table 243. Initialization and Termination user exit parameter list: IMSplex termination (continued)

Offset	Length	Field usage	Description
X'08'	X'08'	Input	IMSplex name

Contents of registers on exit

Register	Contents	
15	Return code	Meaning
	0	Always zero
	All other registers must be restored.	

CSL ODBM Input user exit routine

The CSL ODBM Input user exit routine is called by ODBM CSLDMI API requests. You can use this exit routine, for example, to alter segment search arguments (SSAs), an I/O area, an application interface block (AIB), the PSB name, or the alias name.

The CSL ODBM Input user exit routine is optional.

The CSL ODBM Input user exit routine is driven for the following events:

- All CSLDMI FUNC=ODBMCI requests
- FUNC=ODBMCLIENT APSB requests
- FUNC=ODBMCLIENT APSB preprocessing request

The CSL ODBM Input user exit routine is defined as TYPE=INPUT in the EXITDEF statement in the BPE user exit list PROCLIB member. The user can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS keyword.

The CSL ODBM Input user exit routine is invoked in 31-bit addressing mode (AMODE 31) and should be reentrant.

The CSL ODBM Input user exit can capture the user client ID from any transaction that uses the CSLDMI FUNC=ODBMCI request. The client ID is passed to the exit as a parameter of the ODBM APSB thread token. This information can be associated with the transaction details from the relevant IMS type X'08' log record to create a charge-back profile, an audit trail, or to drive other business processes.

Contents of registers on entry

On entry to the CSL ODBM Input user exit routine, register 1 points to a standard BPE user exit parameter list. The registers contain the following:

Register	Contents
1	Address of BPE User Exit Parameter List (mapped by macro BPEUXPL).
13	Address of 2 pre-chained save areas. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

Parameter list

Field UXPL_EXITPLP in this list contains the address of the CSL ODBM Input user exit routine parameter list (mapped by macro CSLDINX). Field UXPPL_COMPTYPEP in this list points to the character string "ODBM" indicating an ODBM-type address space.

Note: When the function code is 2, all parameter list fields are zero except the following:

- Client ID fields (if available)
- z/OS Resource Recovery Services parent UR token (if available)
- ODBM APSB call thread token
- User-defined request token (if available)
- PSB name
- Alias name

When the function code is 3, all parameter list fields are zero except the following:

- Client ID fields (if available)
- z/OS Resource Recovery Services parent UR token (if available)
- User-defined request token (if available)
- PSB name
- Alias name

Table 244. ODBM Input user exit parameter list

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000002').
X'04'	X'04'	Input	Function code: **1** ODBM CSLDMI FUNC = ODBMCI request **2** ODBM CSLDMIC FUNC = ODBMCLIENT APSB request **3** ODBM CSLDMIC FUNC = ODBMCLIENT APSB preprocessing request
X'08'	X'04'	Input	Length of AIB
X'0C'	X'04'	Input	Address of copy of AIB
X'10'	X'04'	Output	AIB change indicator: X'00'=AIB not modified X'01'=AIB modified. Use modified AIB.
X'14'	X'04'	Input	Length of the client ID (CLIENTIDLEN)
X'18'	X'04	Input	Address of the client ID (CLIENTID)
X'1C'	X'04'	Input	Function code specified on DLIFUNC parameter
X'20'	X'04'	Input	Length of IO area or 0
X'24'	X'04'	Input	Address of copy of IO area or 0

Table 244. ODBM Input user exit parameter list (continued)

Offset	Length	Field usage	Description
X'28'	X'04'	Output	IO area change indicator: X'00'=IO area not modified X'01'=IO area modified. Used modified IO area. **Note:** If the user exit modifies the IO area, then you must set the AIB field AIBOALEN to the length of the IO area to be used on the actual IMS DLI call. You must also indicate to ODBM that the AIB has been modified. On return to ODBM from the exit, if the length specified in field AIBOALEN is greater than the length specified on the IOAREALEN parameter for the CSLDMI call, ODBM will reject the CSLDMI call.
X'2C'	X'04'	Input	Length of SSA1 or 0
X'30'	X'04'	Input	Address of copy of SSA1 or 0
X'34'	X'04'	Output	SSA1 change indicator: X'00'=SSA1 not modified X'01'=SSA1 modified. Use modified SSA1.
X'38'	X'04'	Input	Length of SSA2 or 0
X'3C'	X'04'	Input	Address of copy of SSA2 or 0
X'40'	X'04'	Output	SSA2 change indicator: X'00'=SSA2 not modified X'01'=SSA2 modified. Use modified SSA2.
X'44'	X'04'	Input	Length of SSA3 or 0
X'48'	X'04'	Input	Address of copy of SSA3 or 0
X'4C'	X'04'	Output	SSA3 Change indicator: X'00'=SSA3 not modified X'01'=SSA3 modified. Use modified SSA3.
X'50'	X'04'	Input	Length of SSA4 or 0
X'54'	X'04'	Input	Address of copy of SSA4 or 0
X'58'	X'04'	Output	SSA4 change indicator: X'00'=SSA4 not modified X'01'=SSA4 modified. Use modified SSA4.

Table 244. ODBM Input user exit parameter list (continued)

Offset	Length	Field usage	Description
X'5C'	X'04'	Input	Length of SSA5 or 0
X'60'	X'04'	Input	Address of copy of SSA5 or 0
X'64'	X'04'	Output	SSA5 change indicator: X'00'=SSA5 not modified X'01'=SSA5 modified. Use modified SSA5.
X'68'	X'04'	Input	Length of SSA6 or 0
X'6C'	X'04'	Input	Address of copy of SSA6 or 0
X'70'	X'04'	Output	SSA6 change indicator: X'00'=SSA6 not modified X'01'=SSA6 modified. Use modified SSA6.
X'74'	X'04'	Input	Length of SSA7 or 0
X'78'	X'04'	Input	Address of copy of SSA7 or 0
X'7C'	X'04'	Output	SSA7 change indicator: X'00'=SSA7 not modified X'01'=SSA7 modified. Use modified SSA7.
X'80'	X'04'	Input	Length of SSA8 or 0
X'84'	X'04'	Input	Address of copy of SSA8 or 0
X'88'	X'04'	Output	SSA8 change indicator: X'00'=SSA8 not modified X'01'=SSA8 modified. Use SSA8.
X'8C'	X'04'	Input	Length of SSA9 or 0
X'90'	X'04'	Input	Address of copy of SSA9 or 0
X'94'	X'04'	Output	SSA9 change indicator: X'00'=SSA9 not modified X'01'=SSA9 modified. Use SSA9.
X'98'	X'04'	Input	Length of SSA10 or 0
X'9C'	X'04'	Input	Address of copy of SSA10 or 0
X'A0'	X'04'	Output	SSA10 change indicator: X'00'=SSA10 not modified X'01'=SSA10 modified. Use SSA10.
X'A4'	X'04'	Input	Length of SSA11 or 0
X'A8'	X'04'	Input	Address of copy of SSA11 or 0

Table 244. ODBM Input user exit parameter list (continued)

Offset	Length	Field usage	Description
X'AC'	X'04'	Output	SSA11 change indicator: X'00'=not modified X'01'=SSA11 modified. Use SSA11.
X'B0'	X'04'	Input	Length of SSA12 or 0
X'B4'	X'04'	Input	Address of copy of SSA12 or 0
X'B8'	X'04'	Output	SSA12 change indicator: X'00'=not modified X'01'=SSA12 modified. Use SSA12.
X'BC'	X'04'	Input	Length of SSA13 or 0
X'C0'	X'04'	Input	Address of copy of SSA13 or 0
X'C4'	X'04'	Output	SSA13 change indicator: X'00'=not modified X'01'=SSA13 modified. Use SSA13.
X'C8'	X'04'	Input	Length of SSA14 or 0
X'CC'	X'04'	Input	Address of copy of SSA14 or 0
X'D0'	X'04'	Output	SSA14 change indicator: X'00'=not modified X'01'=SSA14 modified. Use SSA14.
X'D4'	X'04'	Input	Length of SSA15 or 0
X'D8'	X'04'	Input	Address of copy of SSA15 or 0
X'DC'	X'04'	Output	SSA15 change indicator: X'00'=not modified X'01'=SSA15 modified. Use SSA15.
X'E0'	X'10'	Input	RRS parent UR token or 0 (URTOKEN)
X'F0'	X'10'	Input	Context Services private context token or 0 (CTXTOKEN)
X'100'	X'10'	Input	ODBM APSB call thread token or 0 (APSBTOKEN)
X'110'	X'10'	Input	User-defined request token or 0 (RQSTTKN1)
X'120'	X'08'	Input	PSB name (Maximum of 8 characters padded with blanks to the right if fewer than 8 characters)
X'128'	X'04'	Output	PSB name change indicator X'00' = PSB name not modified X'01' = PSB name modified

Table 244. ODBM Input user exit parameter list (continued)

Offset	Length	Field usage	Description
X'12C'	X'04'	Input	Alias name (Maximum of 4 characters padded with blanks to the right if fewer than 4 characters)
X'130'	X'04'	Output	Alias name change indicator X'00' = Alias name not modified X'01' = Alias name modified

Contents of registers on exit

The following table shows the return code values that your user-supplied exit routine can set in register 15 before returning control to IMS, and the meaning of each value.

Table 245. Return code values that your user-supplied exit routine can set in register 15

Register	Contents	
15	Return code:	Meaning:
	0	Continue processing
	4	The user exit has indicated that the PSB schedule request is to be rejected.

CSL ODBM Output user exit routine

The CSL ODBM Output user exit routine is called to view output, such as ODBA call output, going from ODBM to the ODBM client in response to a CSLDMI FUNC=ODBMCI request. The exit also has the ability to modify the output before it is returned to the originator.

The CSL ODBM Output user exit routine is optional

The CSL ODBM Output user exit routine is driven for the following events:

- All CSLDMI FUNC=ODBMCI requests

The CSL ODBM Output user exit routine is defined as TYPE=OUTPUT in the EXITDEF statement in the BPE user exit list PROCLIB member. The user may specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. Refer to the ODBM User Exit List PROCLIB Member for more information on how to define user exit module names.

The exit is invoked in 31-bit addressing mode (AMODE 31) and should be reentrant.

Contents of registers on entry

On entry, the CSL ODBM Output user exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “Standard BPE user exit parameter list” on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the ODBM Output user exit parameter list, which is mapped by macro CSLDOUX.
13	Address of 2 pre-chained save areas. The first save area may be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.

Register	Contents
14	Return address.
15	Entry point of exit routine.

Parameter list

Table 246. ODBM Output user exit parameter list

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **X'1'** ODBM CSLDMI FUNC=ODBMCI request
X'08'	X'04'	Input	Length of AIB
X'0C'	X'04'	Input	Address copy of AIB
X'10'	X'04'	Output	AIB change indicator: **X'00** The AIB was not modified. **X'01** The AIB was modified. Use the modified AIB.
X'14'	X'04'	Input	Length of PCB or 0
X'18'	X'04'	Input	Address of copy of PCB (read only) or 0
X'1C'	X'04'	Input	Function code specified on DLIFUNC parameter
X'20'	X'04'	Input	Length of I/Oarea or 0
X'24'	X'04'	Input	Address of copy of I/O area or 0
X'28'	X'04'	Output	IO area change indicator: **X'00'** The I/O area was not modified. **X'01'** The I/O area was modified. Use the modified I/O area. **Note:** If the user exit modifies the I/O area and the ODBM client that issued CSLDMI is sensitive to the AIB field AIBOAUSE, the user exit must set field AIBOAUSE as appropriate and indicate to ODBM that the AIB has been modified.
X'2C'	X'04'	Input	Length of SSA1 or 0
X'30'	X'04'	Input	Address of copy of SSA1 or 0
X'34'	X'04'	Input	Length of SSA2 or 0
X'38'	X'04'	Input	Address of copy of SSA2 or 0
X'3C'	X'04'	Input	Length of SSA3 or 0
X'40'	X'04'	Input	Address of copy of SSA3 or 0
X'44'	X'04'	Input	Length of SSA4 or 0

Table 246. ODBM Output user exit parameter list (continued)

Offset	Length	Field usage	Description
X'48'	X'04'	Input	Address of copy of SSA4 or 0
X'4C'	X'04'	Input	Length of SSA5 or 0
X'50'	X'04'	Input	Address of copy of SSA5 or 0
X'54'	X'04'	Input	Length of SSA6 or 0
X'58'	X'04'	Input	Address of copy of SSA6 or 0
X'5C'	X'04'	Input	Length of SSA7 or 0
X'60'	X'04'	Input	Address of copy of SSA7 or 0
X'64'	X'04'	Input	Length of SSA8 or 0
X'68'	X'04'	Input	Address of copy of SSA8 or 0
X'6C'	X'04'	Input	Length of SSA9 or 0
X'70'	X'04'	Input	Address of copy of SSA9 or 0
X'74'	X'04'	Input	Length of SSA10 or 0
X'78'	X'04'	Input	Address of copy of SSA10 or 0
X'7C'	X'04'	Input	Length of SSA11 or 0
X'80'	X'04'	Input	Address of copy of SSA11 or 0
X'84'	X'04'	Input	Length of SSA12 or 0
X'88'	X'04'	Input	Address of copy of SSA12 or 0
X'8C'	X'04'	Input	Length of SSA13 or 0
X'90'	X'04'	Input	Address of copy of SSA13 or 0
X'94'	X'04'	Input	Length of SSA14 or 0
X'98'	X'04'	Input	Address of copy of SSA14 or 0
X'9C'	X'04'	Input	Length of SSA15 or 0
X'A0'	X'04'	Input	Address of copy of SSA15 or 0
X'A4'	X'10'	Input	z/OS Resource Recovery Services parent UR token or 0 (URTOKEN)
X'B4'	X'10'	Input	Context Services private context token or 0 (CTXTOKEN)
X'C4'	X'10'	Input	ODBM APSB call thread token or 0 (APSBTOKEN)
X'D4'	X'10'	Input	User defined request token or 0 (RQSTTKN1)
X'E4'	X'04'	Input	Length of client ID (CLIENTIDLEN)
X'E8'	X'04'	Input	Address of client ID (CLIENTID)

Contents of registers on exit

Register	Contents	
15	Return code:	Meaning:
	0	Always zero

Register	Contents
	All other registers must be restored.

CSL ODBM Client Connect and Disconnect user exit routine

This exit is called when a client registers to or de-registers from ODBM.

The CSL ODBM Client Connect and Disconnect user exit routine is optional.

The CSL ODBM Client Connect and Disconnect user exit routine is called for the following events:

- A client issues the CSLDMREG request to indicate that the client is ready to communicate with ODBM.
- A client issues the CSLDMDRG request to indicate that the client is no longer communicating with ODBM.

The CSL ODBM Client Connect and Disconnect user exit routine is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You may specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword.

The CSL ODBM Client Connect and Disconnect user exit routine is invoked in 31-bit addressing mode (AMODE 31) and should be reentrant.

Contents of registers on entry

On entry, the CSL ODBM Client Connect and Disconnect user exit routine must save all registers using the provided SAVEAREA. The registers contain the following:

Register	Contents
1	Address of Standard BPE user exit parameter list (mapped by the BPEUXPL macro).
13	Address of 2 pre-chained saveareas. The first savearea may be used by exit to save registers on entry. The second savearea is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

Parameter list

On entry to the CSL ODBM Client Connect and Disconnect user exit routine, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the ODBM Client Connect and Disconnect user exit parameter list (mapped by macro CSLDCLX). Field UXPL_COMPTYPEP in this list points to the character string "ODBM" indicating an ODBM type address space.

The following tables lists the user exit parameter list for the ODBM Client Connection and ODBM Client Disconnection. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 247. ODBM client connection user exit parameter list: Client Connect

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **X'1'** ODBM registration.
X'08'	X'08'	Input	Client (IMSplex member) name.
X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).

Table 247. ODBM client connection user exit parameter list: Client Connect (continued)

Offset	Length	Field usage	Description
X'12'	X'02'	None	Reserved.
X'14'	X'08'	Input	IMSplex member subtype.
X'1C'	X'04'	None	Reserved.

Table 248. ODBM client connection user exit parameter list: Client Disconnect

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **X'2'** ODBM deregistration.
X'08'	X'08'	Input	Client (IMSplex member) name.
X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
X'12'	X'01'	Input	Flag byte indicates whether the client disconnect is normal or abnormal: **X'80'** Client disconnect is abnormal.
X'13'	X'01'	None	Reserved.
X'14'	X'08'	Input	IMSplex member subtype.
X'1C'	X'04'	None	Reserved.

Contents of registers on exit

Register	Contents	
15	**Return code**	**Meaning**
	0	Always zero
	All other registers must be restored.	

CSL ODBM statistics available through BPE statistics user exit

The BPE Statistics user exit can be used to gather both BPE and ODBM statistics.

This topic describes ODBM statistics that are:

- available to the BPE Statistics user exit when driven from an ODBM address space
- returned on a CSLZQRY FUNC=STATS request directed to the ODBM address space

When the user exit is driven, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the ODBM statistics header. When the CSLZQRY FUNC=STATS request is made, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the ODBM statistics header. The header is mapped by CSLDSTX.

Subsections:

- "CSL ODBM statistics header" on page 587
- "CSL ODBM statistics record CSLDST1" on page 587
- "CSL ODBM statistics record CSLDST2" on page 588

CSL ODBM statistics header

The following table lists the CSL ODBM statistics header. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field. The header is mapped by CSLDSTX.

Table 249. ODBM statistics header

Field name	Offset	Length	Field usage	Description
DSTX_ID	X'00'	X'08'	Input	Eye catcher "CSLDSTX".
DSTX_LEN	X'08'	X'04'	Input	Length of header.
DSTX_PVER	X'0C'	X'04'	Input	Header version number (0000001).
DSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.
DSTX_STATCNT	X'14'	X'04'	Input	Number of statistics areas available for each IMSplex.
DSTX_STATLEN	X'18'	X'04'	Input	Length of all statistics areas for each IMSplex.
DSTX_STATOFF	X'1C'	X'04'	Input	Offset to the statistics area for the first IMSplex (offset from CSLDSTX).
DSTX_DST1OFF	X'20'	X'04'	Input	CSLDST1 offset within the statistics area for each IMSplex.
DSTX_DST2OFF	X'24'	X'04'	Input	CSLDST2 offset within the statistics area for each IMSplex.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

CSL ODBM statistics record CSLDST1

CSLDST1 contains statistics that are related to specific requests processed by ODBM. The following table lists the ODBM statistics record CSLDST1. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 250. ODBM statistics record CSLDST1

Field name	Offset	Length	Field usage	Description
DST1_ID	X'00'	X'08'	Input	Eye catcher "CSLDST1".
DST1_LEN	X'08'	X'04'	Input	Control block length.
DST1_PVER	X'0C'	X'04'	Input	Version number (0000001).
DST1_STATS	X'10'	X'04'	Input	Start of statistics.
DST1_DMREG	X'10'	X'04'	Input	CSLDMREG requests.
DST1_DMDRG	X'14'	X'04'	Input	CSLDMDRG requests.
	X'18'	X'04'	None	Reserved.
	X'1C'	X'04'	None	Reserved.
DST1_DMDRGIN	X'20'	X'04'	Input	Internal Dereg (normal).
DST1_DMDRGIA	X'24'	X'04'	Input	Internal Dereg (abnormal).

Table 250. ODBM statistics record CSLDST1 (continued)

Field name	Offset	Length	Field usage	Description
DST1_DMI	X'28'	X'04'	Input	CSLDMI ODBMCI requests.
DST1_DMICCLNT	X'2C'	X'04'	Input	CSLDMIC ODBMCLIENT requests.
DST1_DMISUSI	X'30'	X'04'	Input	CSLDMI READYSYNCPT requests.
DST1_DMICMIT	X'34'	X'04'	Input	CSLDMI COMMIT requests.
DST1_DMIBACK	X'38'	X'04'	Input	CSLDMI BACKOUT requests.
	X'3C'	X'04'	None	Reserved for CSLDMI.
	X'40'	X'04'	None	Reserved.
	X'44'	X'04'	None	Reserved.
	X'48'	X'04'	None	Reserved.
	X'4C'	X'04'	None	Reserved.
	X'50'	X'04'	None	Reserved.
	X'54'	X'04'	None	Reserved.
	X'58'	X'04'	None	Reserved.
DST1_ZQRY	X'5C'	X'04'	Input	CSLZQRY requests.
DST1_ZSHUT	X'60'	X'04'	Input	CSLZSHUT requests.
	X'64'	X'04'	None	Reserved.
	X'68'	X'04'	None	Reserved.
	X'6C'	X'04'	None	Reserved.
	X'74'	X'04'	None	Reserved.
	X'78'	X'04'	None	Reserved.
	X'7C'	X'04'	None	Reserved.

CSL ODBM statistics record CSLDST2

CSLDST2 contains statistics that are related to specific requests processed by ODBM. The following table lists the ODBM statistics record CSLDST2. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 251. ODBM statistics record CSLDST2

Field name	Offset	Length	Field usage	Description
DST2_ID	X'00'	X'08'	Input	Eye catcher "CSLDST2"
DST2_LEN	X'08'	X'04'	Input	Length of valid data.
DST2_PVER	X'0C'	X'04'	Input	Version number (0000001).
DST2_STATS	X'10'	X'04'	Input	Start of statistics.
DST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
DST2_CLIENTS	X'18'	X'04'	Input	Number of registered clients.

Table 251. ODBM statistics record CSLDST2 (continued)

Field name	Offset	Length	Field usage	Description
	X'1C'	X'04'	None	Reserved.
	X'20'	X'04'	None	Reserved.
	X'24'	X'04'	None	Reserved.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

CSL OM user exit routines

You can write OM user exits to customize and monitor the OM environment. No sample exits are provided.

OM uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends.

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CSL OM client connection user exit

This exit is called when a client registers or deregisters commands with OM. This exit is optional.

This exit is called for the following event:

- A client issues the CSLOMRDY request to indicate that the client is ready to accept commands for processing.

This exit is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 15.2 System Definition*.

The exit is invoked in 31-bit addressing mode (AMODE 31) and should be reentrant.

Contents of registers on entry

Register	Contents
1	Address of the "Standard BPE user exit parameter list" on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the OM Client Connection user exit parameter list, which is mapped by macro CSLOCLX.
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

OM client connection user exit parameter list: Client Connect

The following table lists the user exit parameter list for OM Client Connection. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 252. OM client connection user exit parameter list: Client Connect

Field name	Offset	Length	Field usage	Description
OCLX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OCLX_FUNC	X'04'	X'04'	Input	Function code:: **3** Client ready to process commands.
OCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
OCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
	X'12'	X'02'	None	Reserved.
OCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype.
	X'1C'	X'04'	None	Reserved.

OM client connection user exit parameter list: Client Disconnect

The following table lists the user exit parameter list for OM Client Disconnect. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 253. OM client connection user exit parameter list: Client Disconnect

Offset	Length	Field usage	Description
X'00'	X'04'	Input	Parameter list version number (X'00000001').
X'04'	X'04'	Input	Function code: **2** Client no longer processing commands.
X'08'	X'08'	Input	Client (IMSplex member) name.
X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
X'12'	X'01'	Input	Flag byte indicates whether the client disconnect is normal or abnormal. **X'80'** Client disconnect is abnormal.
X'13'	X'01'	None	Reserved.
X'14'	X'08'	Input	IMSplex member subtype.
X'1C'	X'04'	None	Reserved.

Contents of registers on exit

Register	Contents
15	Return code: **0** The return code must be zero.
	All other registers must be restored.

CSL OM Initialization/termination user exit

This exit enables you to initialize or terminate work areas or control blocks specific to a user-written SPOC application. This exit is not called during OM address space abnormal termination or IMSplex abnormal termination. This exit is optional.

This exit is called for the following events:

- After OM has completed initialization
- After each IMSplex has initialized
- When OM is normally terminating
- When an IMSplex is normally terminating

This exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 15.2 System Definition*.

Subsections:

- "OM init/term user exit parameter list: OM Initialization" on page 591
- "OM init/term user exit parameter list: OM Termination" on page 592
- "OM init/term user exit parameter list: IMSplex Initialization" on page 592
- "OM init/term user exit parameter list: IMSplex Termination" on page 592

Contents of registers on entry

Register	Contents
1	Address of the "Standard BPE user exit parameter list" on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the OM Initialization/termination exit routine parameter list, which is mapped by macro CSLOITX.
13	Address of the first of 2 pre-chained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

This exit is invoked in amode 31 and should be coded as reentrant.

OM init/term user exit parameter list: OM Initialization

The following table lists the user exit parameter list for OM Initialization. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 254. OM init/term user exit parameter list: OM Initialization

Field name	Offset	Length	Field usage	Description
OITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OITX_FINIT	X'04'	X'04'	Input	Function code: **1** OM initialization.

OM init/term user exit parameter list: OM Termination

The following table lists the user exit parameter list for OM Termination. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 255. OM init/term user exit parameter list: OM Termination

Field name	Offset	Length	Field usage	Description
OITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OITX_FTERM	X'04'	X'04'	Input	Function code: **1** OM normal termination.

OM init/term user exit parameter list: IMSplex Initialization

The following table lists the user exit parameter list for IMSplex initialization. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 256. OM init/term user exit parameter list: IMSplex Initialization

Field name	Offset	Length	Field usage	Description
OITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OITX_FPLXINIT	X'04'	X'04'	Input	Function code: **3** IMSplex normal termination.
OITX_IPLEXNM	X'08'	X'08'	Input	IMSplex name.

OM init/term user exit parameter list: IMSplex Termination

The following table lists the user exit parameter list for IMSplex termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 257. OM init/term user exit parameter list: IMSplex Termination

Field name	Offset	Length	Field usage	Description
OITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OITX_FPLXTERM	X'04'	X'04'	Input	Function code: **4** IMSplex normal termination.
OITX_TPLEXNM	X'08'	X'08'	Input	IMSplex name.

Contents of registers on exit

Register	Contents
15	Return code: **0** The return code must be zero.
	All other registers must be restored.

CSL OM input user exit

This exit is called to allow a user to view and manipulate command input from an OM automation client. This exit is optional.

This exit is called for the following event:

- OM receives a command. This exit is called before OM processes the command, which allows the command to be modified or rejected.

This exit is defined as TYPE=INPUT in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB Member topic in *IMS Version 15.2 System Definition*.

Subsection

- "OM input user exit parameter list: Command Input" on page 593

Contents of registers on entry

Register	Contents
1	Address of the "Standard BPE user exit parameter list" on page 485. The UXPL_EXITPLP field in this list contains the address of the OM Input user exit routine parameter list, which is mapped by macro CSLOINX.
13	Address of the first of 2 pre-chained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

This exit is invoked amode 31 and should be coded as reentrant.

OM input user exit parameter list: Command Input

The following table lists the user exit parameter list for command input. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 258. OM input user exit parameter list: Command Input

Field name	Offset	Length	Field usage	Description
OINX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OINX_FUNC	X'04'	X'04'	Input	Function code **1** Command input.
OINX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) where command originated.
OINX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type where command originated.

Table 258. OM input user exit parameter list: Command Input (continued)

Field name	Offset	Length	Field usage	Description
OINX_CMDMOD	X'12'	X'01'	Output	Command input modified field. This field indicates that the exit modified the command input string and that the updated command input should be processed. **4** Command input was modified by the exit. This is the only valid value. All other values are ignored.
	X'13'	X'01'	None	Reserved.
OINX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype where command originated.
OINX_USERID	X'1C'	X'08'	Input	user ID of application where the command originated.
OINX_INPUTLEN	X'24'	X'04'	Input	Length of the command input string. This length does not include 80 bytes for command expansion.
OINX_INPUTPTR	X'28'	X'04'	Input	Address of the command input string. The command input string is followed by 80 blanks that can be used by the exit to expand the command input.
OINX_INMODLEN	X'2C'	X'04'	Output	New length of command input string after being modified by the exit. The exit must set this field if it modifies the command input string. If the exit indicates that the command input string was modified and this field does not contain a value, the command will be rejected.
OINX_ROUTLLEN	X'30'	X'04'	Input	Length of the ROUTE list. If this field is zero, there is no ROUTE list; the default option of routing to all clients was selected.
OINX_ROUTLPTR	X'34'	X'04'	Input	Address of the ROUTE list. The ROUTE list cannot be modified by this exit. The ROUTE list is a list of client names separated by commas. The ROUTE list can contain a single asterisk as a client name, which routes to all clients.
	X'38'	X'10'	None	Reserved.

Contents of registers on exit

Register	Contents
15	Return code: **0** Continue command processing. **4** Reject the command. This return code is ignored unless one of the following conditions is met: • This exit routine is the last routine defined in the exit list for the CSL OM Input exit. • This exit routine sets the byte addressed in the UXPL_CALLNEXTP field of the "Standard BPE user exit parameter list" on page 485 to the value UXPL_CALLNEXTNO.
	All other registers must be restored.

CSL OM output user exit

This exit is called to allow a user to view and manipulate output from OM. This exit is optional.

This exit is called for the following events:

- A command has been processed and is ready to be delivered to the originator of the command. The exit can modify the command response text before the response is delivered.
- When an unsolicited message is received from a client (for example, an IMS control region) using the CSLOMOUT API.

This exit is defined as TYPE=OUTPUT in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 15.2 System Definition*.

This exit is invoked amode 31 and should be reentrant.

Subsections:

- "OM output user exit parameter list: Command Response" on page 596
- "OM output user exit parameter list: Undeliverable Output" on page 597
- "OM output user exit parameter list: Unsolicited Output" on page 598

Contents of registers on entry

Register	Contents
1	Address of the "Standard BPE user exit parameter list" on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the OM Output user exit routine parameter list, which is mapped by macro CSLOOUX.
13	Address of the first of 2 pre-chained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

OM output user exit parameter list: Command Response

The following table lists the user exit parameter list for command response. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 259. OM output user exit parameter list: command response

Field name	Offset	Length	Field usage	Description
OOUX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OOUX_FUNC	X'04'	X'04'	Input	Function code **2** Command response.
OOUX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name that sent the command to OM.
OOUX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type that sent the command to OM.
OOUX_OUTMOD	X'12'	X'01'	Output	Output modified indicator. This field indicates that the command output has been modified. The field should be set to 4 to have OM process the modified command response; otherwise, set the field to 0. • **1** Output was not modified. • **4** Output modified by the exit.
	X'13'	X'01'	None	Reserved.
OOUX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype that sent the command to OM.
OOUX_INPUTLEN	X'1C'	X'04'	Input	Length of the command input, if available.
OOUX_INPUTPTR	X'20'	X'04'	Input	Address of the command input, if available.
OOUX_OUTPTLEN	X'24'	X'04'	Input	Length of the command response.
OOUX_OUTPTPTR	X'28'	X'04'	Input	Address of the command response. Command response output is in XML format wrapped with the tags <imsout>...</imsout>.
OOUX_OUTMDLEN	X'2C'	X'04'	Output	Modified command output length. The exit must set this field if it modifies the command response output. This field must not be greater than the input command response length passed to this exit. If the exit does not set this field appropriately and does modify the command response output, the modified command response output will not be delivered to the client. Instead, the original command response output will be sent to the client.
OOUX_RQTKN1	X'30'	X'10'	Input	Request token 1.
OOUX_RQTKN2	X'40'	X'10'	Input	Request token 2.
OOUX_RETCODE	X'50'	X'04'	Input	Return code being sent to the client.
OOUX_RSNCODE	X'54'	X'04'	Input	Reason code being sent to the client

Table 259. OM output user exit parameter list: command response (continued)

Field name	Offset	Length	Field usage	Description
	X'58'	X'10'	None	Reserved.

OM output user exit parameter list: Undeliverable Output

The following table lists the user exit parameter list for undeliverable output. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 260. OM output user exit parameter list--undeliverable output

Field name	Offset	Length	Field usage	Description
OOUX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OOUX_FUNC	X'04'	X'04'	Input	Function code **3** Undeliverable command response.
OOUX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name sending the command response.
OOUX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type that sending the command response.
OOUX_OUTMOD	X'12'	X'01'	Output	Output modified field. This field indicates that the exit modified the command response string and that the updated command response should be processed. • **0** Output was not modified. • **4** Output modified by the exit. Undeliverable output does not get passed to any client.
	X'13'	X'01'	None	Reserved.
OOUX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype sending the command response.
OOUX_INPUTLEN	X'1c'	X'04'	Input	Length of the command input (if available)
OOUX_INPUTPTR	X'20'	X'04'	Input	Address of the command input (if available)
OOUX_OUTPTLEN	X'24'	X'04'	Input	Length of the command response or 0 if command response not available.
OOUX_OUTPTPTR	X'28'	X'04'	Input	Address of the command response if available. If the client failed to process the command, the client has returned only return/reason codes and no command response. In this case, the command response length field and this field will be zero.

Table 260. OM output user exit parameter list--undeliverable output (continued)

Field name	Offset	Length	Field usage	Description
OOUX_OUTMDLEN	X'2c'	X'04'	Output	Modified command output length. The exit must set this field if it modifies the command response output. This field must not be greater than the input command response length passed to this exit. If the exit does not set this field appropriately and does modify the command response output, the modified command response output will not be delivered to the client. Instead, the original command response output will be sent to the client.
OOUX_RQTKN1	X'30'	X'10'	Input	Request token 1.
OOUX_RQTKN2	X'40'	X'10'	Input	Request token 2.
OOUX_RETCODE	X'50'	X'04'	Input	Return code from client.
OOUX_RSNCODE	X'54'	X'04'	Input	Reason code from client.
	X'58'	X'10'	None	Reserved.

Contents of registers on exit for command response and undeliverable output

Register	Contents
15	Return code: **0** The return code must be zero.
	All other registers must be restored.

OM output user exit parameter list: Unsolicited Output

The following table lists the user exit parameter list for unsolicited output. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 261. OM output user exit parameter list: unsolicited output

Field name	Offset	Length	Field usage	Description
OOUX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
OOUX_FUNC	X'04'	X'04'	Input	Function code **4** Unsolicited output message.
OOUX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name sending the message.
OOUX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type sending the message.

Table 261. OM output user exit parameter list: unsolicited output (continued)

Field name	Offset	Length	Field usage	Description
OOUX_OUTMOD	X'12'	X'01'	Output	Output modified field. This field indicates that the exit modified the output message string and that the updated output should be passed to the client. • **0** Output was not modified. • **4** Output modified by the exit. Unsolicited output does not get passed to any client.
	X'13'	X'01'	None	Reserved.
OOUX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype sending the message.
	X'1C'	X'04'	None	Reserved.
	X'20'	X'04'	None	Reserved.
OOUX_OUTPUTLEN	X'24'	X'04'	Input	Length of the message.
OOUX_OUTPUTPTR	X'28'	X'04'	Input	Address of the message.
OOUX_OUTMDLEN	X'2C'	X'04'	Output	Modified command output length. The exit must set this field if it modifies the command response output. This field must not be greater than the input command response length passed to this exit. If the exit does not set this field appropriately and does modify the command response output, the modified command response output will not be delivered to the client. Instead, the original command response output will be sent to the client.
OOUX_RQTKN1	X'30'	X'10'	Input	Request token 1.
OOUX_RQTKN2	X'40'	X'10'	Input	Request token 2.
	X'50'	X'18'	None	Reserved.

Contents of registers on exit for unsolicited output

Register	Contents	
15	**Return code**	**Meaning**
	00	For function code X'00000004' (unsolicited output message), log the message if the exit modified it, and send the message to all clients that have subscribed.
	04	For function code X'00000004' (unsolicited output message), do not log the message, and do not send the message to any clients. This return code is ignored unless: • The OM Output user exit routine is the last routine that is defined in the exit list for the output exit. • The OM Output user exit routine sets the byte that is pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO.
	08	For function code X'00000004' (unsolicited output message), send the message to all clients that subscribed to receive messages, but do not log the message. This return code is ignored unless: • The OM Output user exit routine is the last routine that is defined in the exit list for the output exit. • The OM Output user exit routine sets the byte that is pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO.
	12	For function code X'00000004' (unsolicited output message), log the message if the exit modified it, but do not send the message to any clients. This return code is ignored unless: • The OM Output user exit routine is the last routine that is defined in the exit list for the output exit. • The OM Output user exit routine sets the byte that is pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO.
	All other registers must be restored.	

CSL OM Security user exit

Use the OM Security user exit to perform security checking during command processing. This exit is given control after the OM Input exit. This exit is optional.

This exit is invoked when the CMDSEC= parameter on the OM procedure is specified as A or E:

A
Both this exit and RACF (or equivalent) are used for OM command security

E
Only this exit is called for OM command security

This exit is defined as TYPE=SECURITY in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 15.2 System Definition*.

This exit is invoked amode 31 and should be reentrant.

Contents of registers on entry

Register	Contents
1	Address of the “Standard BPE user exit parameter list” on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the OM Security user exit routine parameter list, which is mapped by macro CSLOSCX.
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

OM security user exit parameter list

The following table lists the user exit parameter list for the security user exit. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 262. OM security user exit parameter list

Field name	Offset	Length	Field usage	Description
OSCX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000002').
OSCX_FUNC	X'04'	X'04'	Input	Function code **1** Perform user command security checking.
OSCX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name that sent the command to OM.
OSCX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type that sent the command to OM.
	X'12'	X'02'	None	Reserved.
OSCX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype that sent the command to OM.
OSCX_USERID	X'1C'	X'08'	Input	User ID of application where the command originated.
OSCX_VERB	X'24'	X'10'	Input	Command verb in short form.
OSCX_KEYWORD	X'34'	X'10'	Input	Primary keyword (resource type).
OSCX_INPUTLEN	X'44'	X'04'	Input	Length of the command input string.
OSCX_INPUTPTR	X'48'	X'04'	Input	Address of the command input string.

Table 262. OM security user exit parameter list (continued)

Field name	Offset	Length	Field usage	Description
OSCX_SECCODE	X'4C'	X'04'	Input	Decoded security code. Only valid when the CMDSEC= parameter on the OM procedure is specified as A. • X'00000000': RACF security permits command. • X'00000004': RACF security was not requested. • X'00000008': RACF security requested, but RACF is not available. • X'0000000C': Resource name or class name not defined to RACF. • X'00000010': Command not protected by RACF. • X'00000014': User ID is not authorized for the command. • X'00000018': Class data space deleted or cannot be accessed. • X'0000001C': User ID is not defined.
OSCX_RACRTCAL	X'84'	X'01'	Input	Indicates the last RACROUTE service call that is made by CSLOSEC0. The values in the SAF RC, RACF RC, and RACF reason codes are the values that are returned from this call. OSCX_RACRTCAL can take on the following values: **OSCX_NOCALL** **0** The RACROUTE call was not issued (CMDSEC=E only) **OSCX_RRCREATE** **1** The last RACROUTE call was RACROUTE REQUEST=VERIFY **OSCX_RRFASTAU** **2** The last RACROUTE call was RACROUTE REQUEST=FASTAUTH **OSCX_BLANK** **3** The RACROUTE call was not issued due to blank userid (CMDSEC=A only)
OSCX_SAFCODE	X'50'	X'04'	Input	Security Authorization Facility (SAF) return code. This is only valid when CMDSEC=A is specified.
OSCX_RETCODE	X'54'	X'04'	Input	RACF return code. This is only valid when CMDSEC=A is specified.

Table 262. OM security user exit parameter list (continued)

Field name	Offset	Length	Field usage	Description
OSCX_RSNCODE	X'58'	X'04'	Input	RACF reason code. This is only valid when CMDSEC=A is specified.
OSCX_USERDATA	X'5C'	X'20'	Output	User data. This data is encapsulated by the <userdata>tags in the <cmdsecerr> section of the XML output, if this exit has rejected the command. This user data can contain alphanumeric characters (A-Z, 0-9), or printable characters (not case sensitive), with the exception of the characters &, <, and >. OM will convert any invalid data placed in this field to periods (.) before sending the XML output to the client.
OSCX_ROUTLEN	X'7C'	X'04'	Input	The length of the ROUTE list. If this value is zero (0), no route list exists. The command was routed to all command processing clients that were Ready or Registered.
OSCX_ROUTLPTR	X'80'	X'04'	Input	The address of the ROUTE list. You cannot use this exit to modify the ROUTE list.
	X'85'	X'07'	None	Reserved.

Contents of registers on exit

Register	Contents
15	Return code: **0** Accept the command for processing **4** Reject the command due to an unauthorized user ID. This return code is ignored unless the exit routine meets one of the following criteria: • The exit routine is the last routine defined in the exit list for the security exit. • The exit routine sets the byte addressed by the UXPL_CALLNEXTP field of the "Standard BPE user exit parameter list" on page 485 to the value UXPL_CALLNEXTNO.
	All other registers must be restored.

CSL OM statistics available through BPE statistics user exit

The BPE Statistics user exit can be used to gather both BPE and OM statistics.

This topic describes OM statistics that are:

- available to the BPE Statistics user exit when driven from an OM address space
- returned on a CSLZQRY FUNC=STATS request directed to the OM address space

When the user exit is driven, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the OM statistics header. When the CSLZQRY FUNC=STATS request is made, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the OM statistics header. The header is mapped by CSLOSTX.

Subsections:

CSL OM statistics header

The following table lists the OM statistics header. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 263. OM statistics header

Field name	Offset	Length	Field usage	Description
OSTX_ID	X'00'	X'08'	Input	Eye catcher "CSLOSTX".
OSTX_LEN	X'08'	X'04'	Input	Length of header.
OSTX_PVER	X'0C'	X'04'	Input	Header version number (X'0000001').
OSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.
OSTX_STATCNT	X'14'	X'04'	Input	Number of statistics areas available for each IMSplex.
OSTX_STATLEN	X'18'	X'04'	Input	Length of all statistics areas for each IMSplex.
OSTX_STATOFF	X'1C'	X'04'	Input	Offset to statistics area for first IMSplex. This is the offset from the beginning of CSLOSTX. The offset points to the CSLOST1 area for the first IMSplex.
OSTX_OST1OFF	X'20'	X'04'	Input	Offset to the OM request statistics record for activity performed by OM requests (mapped by macro CSLOST1). The offset is from the start of the statistics area for this IMSplex. Refer to the next table for a description of the OM Request statistics record.
OSTX_OST2OFF	X'24'	X'04'	Input	Offset to OM IMSplex statistics record for activity performed by OM for an IMSplex (mapped by macro CSLOST2). The offset is from the start of the statistics area for this IMSplex. Refer to Table 265 on page 606 for a description of the OM IMSplex statistics record.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

CSL OM statistics record CSLOST1

CSLOST1 contains statistics related to specific requests and commands that are processed by OM. The following table lists the OM statistics record CSLOST1. Included are the field names, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 264. OM statistics record CSLOST1

Field name	Offset	Length	Field usage	Description
OST1_ID	X'00'	X'08'	Input	Eye catcher "CSLOST1".

Table 264. OM statistics record CSLOST1 (continued)

Field name	Offset	Length	Field usage	Description
OST1_LEN	X'08'	X'04'	Input	Length of valid data.
OST1_PVER	X'0C'	X'04'	Input	Statistics version number (X'00000001').
OST1_OMREG	X'10'	X'04'	Input	Number of CSLOMREG requests.
OST1_OMRDY	X'14'	X'04'	Input	Number of CSLOMRDY requests.
	X'18'	X'04'	None	Reserved.
OST1_OMDRG	X'1C'	X'04'	Input	Number of CSLOMDRG requests.
OST1_OMDRGIN	X'20'	X'04'	Input	Number of internal deregister (normal term) requests.
OST1_OMDRGIA	X'24'	X'04'	Input	Number of internal deregister (abnormal term) requests.
OST1_OMICMD	X'28'	X'04'	Input	Number of CSLOMI command requests.
OST1_OMIQRY	X'2C'	X'04'	Input	Number of CSLOMI query requests.
	X'30'	X'04'	None	Reserved.
	X'34'	X'04'	None	Reserved.
	X'38'	X'04'	None	Reserved.
	X'3C'	X'04'	None	Reserved.
OST1_OMCMD	X'40'	X'04'	Input	Number of CSLOMCMD requests.
OST1_OMQRYCLN	X'44'	X'04'	Input	Number of CSLOMQRY client requests.
OST1_OMQRYSYN	X'48'	X'04'	Input	Number of CSLOMQRY syntax requests.
	X'4C'	X'04'	None	Reserved.
	X'50'	X'04'	None	Reserved.
	X'54'	X'04'	None	Reserved.
	X'58'	X'04'	None	Reserved.
OST1_OMRSP	X'5C'	X'04'	Input	Number of CSLOMRSP requests.
OST1_OMOUT	X'60'	X'04'	Input	Number of CSLOMOUT requests.
	X'64'	X'04'	None	Reserved.
	X'68'	X'04'	None	Reserved.
	X'6C'	X'04'	None	Reserved.
	X'70'	X'04'	None	Reserved.
	X'74'	X'04'	None	Reserved.
OST1_ZQRY	X'78'	X'04'	Input	Number of CSLZQRY requests.
OST1_ZSHUT	X'7C'	X'04'	Input	Number of CSLZSHUT requests.
	X'80'	X'04'	None	Reserved.
	X'84'	X'04'	None	Reserved.

Table 264. OM statistics record CSLOST1 (continued)

Field name	Offset	Length	Field usage	Description
	X'88'	X'04'	None	Reserved.
OST1_QRYIPLX	X'8C'	X'04'	Input	Number of QRY IMSPLEX commands.
	X'90'	X'04'	None	Reserved.
	X'94'	X'04'	None	Reserved.
	X'98'	X'04'	None	Reserved.
	X'9C'	X'04'	None	Reserved.
	X'A0'	X'04'	None	Reserved.
	X'A4'	X'04'	None	Reserved.
	X'A8'	X'04'	None	Reserved.
	X'AC'	X'04'	None	Reserved.

CSL OM statistics record CSLOST2

CSLOST2 contains statistics that are related to an IMSplex, but not to a specific request or command. The following table lists the OM statistics record CSLOST2. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 265. OM statistics record CSLOST2

Field name	Offset	Length	Field usage	Description
OST2_ID	X'00'	X'08'	Input	Eye catcher "CSLOST2".
OST2_LEN	X'08'	X'04'	Input	Length of valid data.
OST2_PVER	X'0C'	X'04'	Input	Parameter list version number (X'00000001').
OST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
OST2_CLIENTS	X'18'	X'04'	Input	Number of active clients in the IMSplex.
OST2_CMDTOUT	X'1C'	X'04'	Input	Number of times a command was timed out.
OST2_UNDELIV	X'20'	X'04'	Input	Number of times a command response output could not be returned to the client.
	X'24'	X'04'	None	Reserved.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.
	X'30'	X'04'	None	Reserved.
	X'34'	X'04'	None	Reserved.
	X'38'	X'04'	None	Reserved.
	X'3C'	X'04'	None	Reserved.
	X'40'	X'04'	None	Reserved.
	X'44'	X'04'	None	Reserved.

Table 265. OM statistics record CSLOST2 (continued)

Field name	Offset	Length	Field usage	Description
	X'48'	X'04'	None	Reserved.
	X'50'	X'04'	None	Reserved.
	X'54'	X'04'	None	Reserved.
	X'58'	X'04'	None	Reserved.

Related reference

"BPE Statistics user-supplied exit routine" on page 511
The BPE Statistics user-supplied exit routine is called at regular intervals during the life of a BPE address space, and a final time at normal address shutdown, to gather address-space related statistics.

CSL RM user exit routines

You can write RM user exits to customize and monitor the RM environment. No sample exits are provided.

RM uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends.

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

CSL RM client connection user exit

This exit is called when a client connects (registers) to RM or disconnects (deregisters) from RM. This exit is optional.

This exit is called for the following events:

- After a client has successfully connected to RM.
- After a client has successfully disconnected normally or abnormally from RM.

Subsections:

- "RM client connection user exit parameter list: Client Connect" on page 608
- "RM client connection user exit parameter list: Client Disconnect" on page 608

Contents of registers on entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Client Connection exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the RM Client Connection user exit parameter list, which is mapped by macro CSLRCLX. Field UXPL_COMPTYPEP in this list points to the character string "RM" indicating an RM address space.

This exit is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are called in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the RM BPE user exit List PROCLIB member information in *IMS Version 15.2 System Definition*.

This exit is invoked amode 31 and should be reentrant.

RM client connection user exit parameter list: Client Connect

The following table lists the user exit parameter list for client connect. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 266. RM client connection user exit parameter list: Client Connect

Field name	Offset	Length	Field usage	Description
RCLX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
RCLX_FUNC	X'04'	X'04'	Input	Function code **1** Client Connect.
RCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
RCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
	X'12'	X'02'	None	Reserved.
RCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype
	X'1C'	X'04'	None	Reserved.

RM client connection user exit parameter list: Client Disconnect

The following table lists the user exit parameter list for client disconnect. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 267. RM client connection user exit parameter list: Client Disconnect

Field name	Offset	Length	Field usage	Description
RCLX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
RCLX_FUNC	X'04'	X'04'	Input	Function code **2** Client Disconnect.
RCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
RCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
RCLX_FLAG1	X'12'	X'01'	Input	Flag byte indicates whether the client disconnect is normal or abnormal. X'80' Client disconnect is abnormal.
	X'13'	X'01'	None	Reserved.

Table 267. RM client connection user exit parameter list: Client Disconnect (continued)

Field name	Offset	Length	Field usage	Description
RCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype
	X'1C'	X'08'	None	Reserved.

Contents of registers on exit

Register	Contents	
15	**Return code**	**Meaning**
	0	Always zero
	All other registers must be restored.	

CSL RM initialization/termination user exit

The CSL RM initialization/termination user exit is called when an IMSplex or RM has initialized, completed initialization, or terminated normally.

This exit is not called during RM address space abnormal termination or IMSplex abnormal termination. This exit is optional.

This exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are called in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the RM BPE user exit List PROCLIB member information in *IMS Version 15.2 System Definition*.

This exit is invoked amode 31 and should be reentrant.

Subsections:

- “RM init/term user exit parameter list: RM Initialization” on page 610
- “RM init/term user exit parameter list: RM Termination” on page 610
- “RM init/term user exit parameter list: IMSplex Initialization” on page 610
- “RM init/term user exit parameter list: IMSplex Termination” on page 610

Contents of registers on entry

Register	Contents
1	Address of the “Standard BPE user exit parameter list” on page 485. The UXPL_EXITPLP field in this parameter list contains the address of the RM Initialization/Termination user exit parameter list, which is mapped by macro CSLRITX.
13	Address of the first of 2 pre-chained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

RM init/term user exit parameter list: RM Initialization

The following table lists the user exit parameter list for RM initialization. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 268. RM init/term user exit parameter list: RM Initialization

Field name	Offset	Length	Field usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
RITX_FUNC	X'04'	X'04'	Input	Function code: **1** RM initialization

RM init/term user exit parameter list: RM Termination

The following table lists the user exit parameter list for RM termination. Included are the field name, offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 269. RM init/term user exit parameter list: RM Termination

Field name	Offset	Length	Field usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
RITX_FTERM	X'04'	X'04'	Input	Function code **2** RM normal termination

RM init/term user exit parameter list: IMSplex Initialization

The following table lists the user exit parameter list for IMSplex initialization. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 270. RM init/term user exit parameter list: IMSplex Initialization

Field name	Offset	Length	Field usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
RITX_FPLXINIT	X'04'	X'04'	Input	Function code **3** IMSplex normal initialization
RITX_IPLEXNM	X'08'	X'08'	Input	IMSplex name.
RITX_ISTRNM	X'10'	X'10'	Input	Resource structure name.

RM init/term user exit parameter list: IMSplex Termination

The following table lists the user exit parameter list for IMSplex termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 271. RM init/term user exit parameter list: IMSplex Termination

Field name	Offset	Length	Field usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').

Table 271. RM init/term user exit parameter list: IMSplex Termination (continued)

Field name	Offset	Length	Field usage	Description
RITX_FUNC	X'04'	X'04'	Input	Function code **4** IMSplex normal termination
RITX_TPLEXNM	X'08'	X'08'	Input	IMSplex name.
RITX_TSTRNM	X'10'	X'10'	Input	Resource structure name.

Contents of registers on exit

Register	Contents
15	Return code: **0** The return code for this exit routine must be zero.
	All other registers must be restored.

CSL RM statistics available through BPE statistics user exit

The BPE Statistics user exit can be used to gather both BPE and RM statistics.

The following describes the RM statistics that are available to the BPE Statistics user exit and are returned on a CSLZQRY FUNC=STATS request directed to the RM address space. When the user exit is called, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the RM statistics header. When the CSLZQRY FUNC=STATS request is called, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the RM statistics header. The header is mapped by CSLRSTX.

Subsections:

- "CSL RM statistics header" on page 611
- "CSL RM statistics record CSLRST1" on page 612
- "CSL RM statistics record CSLRST2" on page 613
- "CSL RM statistics record CSLRST3" on page 614

CSL RM statistics header

The following table lists the RM statistics header. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 272. RM statistics header

Field name	Offset	Length	Field usage	Description
RSTX_ID	X'00'	X'08'	Input	Eye catcher "CSLRSTX".
RSTX_LEN	X'08'	X'04'	Input	Length of header.
RSTX_PVER	X'0C'	X'04'	Input	Header version number (X'0000001').
RSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.

Table 272. RM statistics header (continued)

Field name	Offset	Length	Field usage	Description
RSTX_STATCNT	X'14'	X'04'	Input	Number of statistics areas available for each IMSplex.
RSTX_STATLEN	X'18'	X'04'	Input	Length of all statistics areas for each IMSplex.
RSTX_STATOFF	X'1C'	X'04'	Input	Offset to statistics area for first IMSplex. This is the offset from the beginning of CSLRSTX. The offset points to the CSLRST1 area.
RSTX_RST1OFF	X'20'	X'04'	Input	Offset to the RM request statistics record for activity performed by RM requests (mapped by macro CSLRST1). The offset is from the start of the statistics area for this IMSplex. Refer to the next table for a description of the RM request statistics record.
RSTX_RST2OFF	X'24'	X'04'	Input	Offset to RM IMSplex statistics record for activity performed by RM for an IMSplex (mapped by macro CSLRST2). The offset is from the start of the statistics area for this IMSplex. Refer to Table 274 on page 613 for a description of the RM IMSplex statistics record.
RSTX_RST3CNT	X'28'	X'04'	Input	Number of CSLRST3 RM statistics areas (0 if none).
RSTX_RST3OFF	X'2C'	X'04'	Input	Offset from the start of the CSLRSTX area to the first CSLRST3 RM statistics area (0 if none). CSLRST3 areas are contiguous in storage. From the first one, add the value in field RST3_LEN to get to the next one. See “CSL RM statistics record CSLRST3” on page 614.
	X'30'	X'10'	None	Reserved.

CSL RM statistics record CSLRST1

CSLRST1 contains statistics that are related to specific requests processed by RM. The following table lists the RM statistics record CSLRST1. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 273. RM statistics record CSLRST1

Field name	Offset	Length	Field usage	Description
RST1_ID	X'00'	X'08'	Input	Eye catcher "CSLRST1".
RST1_LEN	X'08'	X'04'	Input	Length of valid data.
RST1_PVER	X'0C'	X'04'	Input	Parameter list version number (X'00000001').
RST1_RMUPD	X'10'	X'04'	Input	Number of CSLRMUPD FUNC=UPDATE requests.

Table 273. RM statistics record CSLRST1 (continued)

Field name	Offset	Length	Field usage	Description
RST1_RMQRY	X'14'	X'04'	Input	Number of CSLRMQRY FUNC=QUERY requests.
RST1_RMDEL	X'18'	X'04'	Input	Number of CSLRMDEL FUNC=DELETE requests.
	X'1C'	X'04'	None	Not used.
RST1_RMREG	X'20'	X'04'	Input	Number of CSLRMREG FUNC=REGISTER requests.
RST1_RMDRG	X'24'	X'04'	Input	Number of CSLRMDRG FUNC=DEREGISTER requests.
RST1_RMDRGIN	X'28'	X'04'	Input	Number of internal deregister requests for client normal termination.
RST1_RMDRGIA	X'2C'	X'04'	Input	Number of internal deregister requests for client abnormal termination.
	X'30'	X'10'	Input	Not used.
RST1_RMPRCI	X'40'	X'04'	Input	Number of CSLRMPRI FUNC=INITIATE initiate IMSplex-wide process requests.
RST1_RMPRCT	X'44'	X'04'	Input	Number of CSLRMPRT FUNC=TERMINATE terminate IMSplex-wide process requests.
RST1_RMPRCS	X'48'	X'04'	Input	Number of CSLRMPRS FUNC=PROCESS IMSplex-wide step requests.
RST1_RMPRCR	X'4C'	X'04'	Input	Number of CSLRMPRR FUNC=RESPOND IMSplex-wide step response requests.
RST1_ZQRY	X'50'	X'04'	Input	Number of CSLZQRY requests.
	X'54'	X'04'	None	Not used
RST1_ZSHUT	X'58'	X'04'		Number of CSLZSHUT requests
RST1_QRYSTR	X'5C'	X'04'		Number of QRY STRUC commands
	X'60'	X'20'	None	Not used

CSL RM statistics record CSLRST2

CSLRST2 contains statistics that are related to an IMSplex, but not to specific requests. The following table lists the RM statistics record CSLRST2. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 274. RM statistics record CSLRST2

Field name	Offset	Length	Field usage	Description
RST2_ID	X'00'	X'08'	Input	Eye catcher "CSLRST2"
RST2_LEN	X'08'	X'04'	Input	Length of valid data
RST2_PVER	X'0C'	X'04'	Input	Parameter list version number (X'00000001')

Table 274. RM statistics record CSLRST2 (continued)

Field name	Offset	Length	Field usage	Description
RST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name
blank	X'18'	X'08'	None	Not used
RST2_STRNAME	X'20'	X'10'	Input	Resource structure name
RST2_STRVER	X'30'	X'08'	Input	Resource structure version
RST2_CQSID	X'38'	X'08'	Input	CQS ID
RST2_CLIENTS	X'40'	X'04'	Input	Number of registered clients
RST2_CREATES	X'44'	X'04'	Input	Number of resource creates
RST2_UPDATES	X'48'	X'04'	Input	Number of resource updates
RST2_DELETES	X'4C'	X'04'	Input	Number of resource deletes
blank	X'50'	X'40'	None	Not used

CSL RM statistics record CSLRST3

CSLRST3 contains statistics that are related to an IMSRSC repository. There is one CSLRST3 per active repository to which RM is connected. Locate the first CSLRST3 area by adding the value in field RSTX_RST3OFF to the address of the start of the CSLRSTX area. Locate the next CSLRST3 area by adding the value in field RST3_LEN to the address of the first CSLRST3 area. The number of CSLRST3 areas is in field RSTX_RST3CNT. The CSLRST3 statistics are per repository, not per IMSplex, and are separate from the IMSplex statistics described by fields RSTX_STATCNT and RSTX_STATLEN.

RM can dynamically connect to and disconnect from repositories. Therefore, the number of CSLRST3 areas passed to the BPE Statistics user-supplied exit routine might vary from one call to the next.

All statistics fields in CSLRST3 are cumulative since the time RM connected to the repository. Unless otherwise noted, all time value fields are in microseconds.

The following table lists the RM statistics record CSLRST3. Included are the offset value and length of each field, how the field is used, and a brief description of the field.

Table 275. RM statistics record CSLRST3

Field name	Offset	Length	Field usage	Description
RST3_ID	X'00'	X'08'	Input	Eye catcher "CSLRST3"
RST3_LEN	X'08'	X'04'	Input	Length of CSLRST3 data
RST3_PVER	X'0C'	X'04'	Input	CSLRST3 version number (X'00000001')
RST3_REPONAME	X'10'	X'2C'	Input	Repository name
RST3_REPOTYPE	X'3C'	X'01'	Input	Repository type X'80' = IMSRSC repository
	X'3D'	X'13'	None	Reserved
RST3_RPUPD	X'50'	X'08'	Input	Count of CSLRPUPD requests
RST3_RPQRY	X'58'	X'08'	Input	Count of CSLRPQRY requests
RST3_RPDEL	X'60'	X'08'	Input	Count of CSLRPDEL requests
	X'68'	X'40'	None	Reserved

Table 275. RM statistics record CSLRST3 (continued)

Field name	Offset	Length	Field usage	Description
RST3_LOCKMBR	X'A8'	X'08'	Input	Member lock elapsed time
RST3_LOCKMBRN	X'B0'	X'08'	Input	Number of member lock requests
RST3_LOCKNMLS	X'B8'	X'08'	Input	Name list lock elapsed time
RST3_LOCKNMLSN	X'C0'	X'08'	Input	Number of name list lock requests
RST3_LOCKGEN	X'C8'	X'08'	Input	Generic lock elapsed time
RST3_LOCKGENN	X'D0'	X'08'	Input	Number of generic lock requests
RST3_LOCKLIST	X'D8'	X'08'	Input	List lock elapsed time
RST3_LOCKLISTN	X'E0'	X'08'	Input	Number of list lock requests
RST3_GETMBR	X'E8'	X'08'	Input	STARTMBR REQ=GET elapsed time
RST3_GETMBRN	X'F0'	X'08'	Input	Number of get member requests
RST3_TGETMBR	X'F8'	X'08'	Input	Total elapsed get member time (STARTMBR to ENDMBR or CANCELMBR)
RST3_PUTMBR	X'100'	X'08'	Input	STARTMBR REQ=PUT elapsed time
RST3_PUTMBRNN	X'108'	X'08'	Input	Number of put member requests
RST3_TPUTMBR	X'110'	X'08'	Input	Total elapsed put member time (STARTMBR to ENDMBR or CANCELMBR)
RST3_GETDATA	X'118'	X'08'	Input	Get data elapsed time
RST3_GETDATAN	X'120'	X'08'	Input	Number of get data requests
RST3_PUTDATA	X'128'	X'08'	Input	Put data elapsed time
RST3_PUTDATAN	X'130'	X'08'	Input	Number of put data requests
RST3_STALIST	X'138'	X'08'	Input	Start list elapsed time
RST3_STALISTN	X'140'	X'08'	Input	Number of start list requests
RST3_GETLIST	X'148'	X'08'	Input	Get list elapsed time
RST3_GETLISTN	X'150'	X'08'	Input	Number of get list requests
RST3_STAUOW	X'158'	X'08'	Input	Start UOW elapsed time
RST3_STAUOWN	X'160'	X'08'	Input	Number of start UOW requests
RST3_CMTUOW	X'168'	X'08'	Input	Commit UOW elapsed time
RST3_CMTUOWN	X'170'	X'08'	Input	Number of commit UOW requests
RST3_ABTUOW	X'178'	X'08'	Input	Abort UOW elapsed time
RST3_ABTUOWN	X'180'	X'08'	Input	Number of abort UOW requests
RST3_EDIT	X'188'	X'08'	Input	Edit member elapsed time
RST3_EDITN	X'190'	X'08'	Input	Number of edit member requests
RST3_TEDIT	X'198'	X'08'	Input	Total elapsed EDIT time
RST3_END	X'1A0'	X'08'	Input	End session elapsed time

Table 275. RM statistics record CSLRST3 (continued)

Field name	Offset	Length	Field usage	Description
RST3_ENDN	X'1A8'	X'08'	Input	Number of end session requests
RST3_CANCEL	X'1B0'	X'08'	Input	Cancel session elapsed time
RST3_CANCELN	X'1B8'	X'08'	Input	Number of cancel session requests
RST3_SAVE	X'1C0'	X'08'	Input	Save session elapsed time
RST3_SAVEN	X'1C8'	X'08'	Input	Number of save session requests
RST3_VIEW	X'1D0'	X'08'	Input	View member elapsed time
RST3_VIEWN	X'1D8'	X'08'	Input	Number of view member requests
RST3_BROWSE	X'1E0'	X'08'	Input	Browse member elapsed time
RST3_BROWSEN	X'1E8'	X'08'	Input	Number of browse member requests
	X'1F0'	X'80'	None	Reserved

Related reference

"BPE Statistics user-supplied exit routine" on page 511
The BPE Statistics user-supplied exit routine is called at regular intervals during the life of a BPE address space, and a final time at normal address shutdown, to gather address-space related statistics.

BPE-based CSL SCI user exit routines

SCI user exits allow you to customize and monitor the SCI environment. They are written and supplied by the user. No sample exits are provided.

SCI uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends.

Related reference

"BPE user-supplied exit routine interfaces and services" on page 485
BPE gives you the ability to externally specify the user exit routine interfaces and services to be called for a particular exit routine type using EXITDEF statements in BPE user exit PROCLIB members.

"CSL SCI IMSplex member exit routines" on page 643
This topic describes the exits that SCI can drive in the address space of a registered IMSplex member.

CSL SCI Client Connection user exit

This exit is called when a client connects (registers) or disconnects (deregisters) from SCI. It is also called when a client issues the CSLSCRDY (ready) and the CSLSCQSC (quiesce) requests. This exit is optional.

This exit is called for the following events:

- After a client has successfully connected to SCI.
- After a client has successfully completed the Ready request to SCI.
- After a client has successfully completed the Quiesce request to SCI.
- After a client has successfully disconnected normally or abnormally from SCI.

This exit is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define

user exit module names, see the SCI BPE user exit list PROCLIB member information in *IMS Version 15.2 System Definition*.

This exit is invoked amode 31 and should be reentrant.

Contents of registers on entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Client Connection exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the SCI Client Connection user exit parameter list, which is mapped by macro CSLSCLX. Field UXPL_COMPTYPEP in this list points to the character string "SCI", indicating an SCI address space.

The following sections describe the following user exit parameter lists for SCI:

- client connection
- client disconnect
- client ready
- client quiesce

Subsection:

- "SCI client connection user exit parameter list" on page 617

SCI client connection user exit parameter list

The following table lists the user exit parameter list for SCI client connection. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 276. SCI client connection user exit parameter list

Field name	Offset	Length	Field usage	Description
SCLX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
SCLX_FUNC	X'04'	X'04'	Input	Function code: **1** Client connect. **2** Client disconnect. **3** Client ready. **4** Client quiesce.
SCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
SCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).

Table 276. SCI client connection user exit parameter list (continued)

Field name	Offset	Length	Field usage	Description
SCLX_FLAG1	X'12'	X'01'	Input	Flag byte: **X'80'** Client disconnect is abnormal. **X'40'** Client is authorized.
	X'13'	X'01'	None	Reserved.
SCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype.
SCLX_MBRVSN	X'1C'	X'04'	Input	Member version number.
SCLX_JOBNAME	X'20'	X'08'	Input	Member jobname.
SCLX_USERID	X'28'	X'08'	Input	Member user ID.
SCLX_OSNAME	X'30'	X'08'	Input	Name of the member's operating system.
SCLX_SCITOKEN	X'38'	X'16'	Input	Member SCI token.
	X'48'	X'04'	None	Reserved.
	X'4C'	X'04'	None	Reserved.

Contents of registers on exit

Register	Contents	
15	**Return code**	**Meaning**
	0	Always zero
	All other registers must be restored.	

CSL SCI Initialization/termination user exit

This exit is called during SCI address space initialization, IMSplex initialization, SCI address space normal termination, or IMSplex normal termination. This exit is not called during SCI address space abnormal termination or IMSplex abnormal termination. This exit is optional.

This exit is called for the following events:

- After SCI has completed initialization
- After each IMSplex has initialized
- When SCI is terminating normally
- When an IMSplex is terminating normally

This exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the SCI BPE user exit list PROCLIB member information in *IMS Version 15.2 System Definition*.

This exit is invoked amode 31 and should be reentrant.

Subsections:

- “SCI init/term user exit parameter list: SCI Initialization” on page 619

- “SCI init/term user exit parameter list: SCI Termination” on page 619
- “SCI init/term user exit parameter list: IMSplex Initialization” on page 619
- “SCI init/term user exit parameter list: IMSplex Termination” on page 620

Contents of registers on entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Initialization/Termination exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the SCI Initialization/Termination user exit parameter list, which is mapped by macro CSLSITX. Field UXPL_COMPTYPEP in this list points to the character string "SCI" indicating an SCI address space.

SCI init/term user exit parameter list: SCI Initialization

The following table lists the user exit parameter list for SCI initialization. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 277. SCI init/term user exit parameter list: SCI Initialization

Field name	Offset	Length	Field usage	Description
SITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
SITX_FUNC	X'04'	X'04'	Input	Function code **1** SCI initialization.

SCI init/term user exit parameter list: SCI Termination

The following table lists the user exit parameter list for SCI termination. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 278. SCI init/term user exit parameter list: SCI Termination

Field name	Offset	Length	Field usage	Description
SITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
SITX_FUNC	X'04'	X'04'	Input	Function code **2** SCI normal termination.

SCI init/term user exit parameter list: IMSplex Initialization

The following table lists the user exit parameter list for IMSplex initialization. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 279. SCI init/term user exit parameter list: IMSplex Initialization

Field name	Offset	Length	Field usage	Description
SITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
SITX_FUNC	X'04'	X'04'	Input	Function code **3** IMSplex normal initialization.
SITX_IPLEXNM	X'08'	X'08'	Input	IMSplex name.

SCI init/term user exit parameter list: IMSplex Termination

The following table lists the user exit parameter list for IMSplex termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 280. SCI init/term user exit parameter list: IMSplex Termination

Field name	Offset	Length	Field usage	Description
SITX_PVER	X'00'	X'04'	Input	Parameter list version number (X'00000001').
SITX_FUNC	X'04'	X'04'	Input	Function code **4** IMSplex normal termination.
SITX_IPLEXNM	X'08'	X'08'	Input	IMSplex name.

Contents of registers on exit

Register	Contents	
15	**Return code**	**Meaning**
	0	Always zero
	All other registers must be restored.	

CSL SCI statistics available through BPE statistics user exit

The BPE Statistics user exit can be used to gather both BPE and SCI statistics.

The following describes SCI statistics that are available to the BPE Statistics User Exit and are returned on a CSLZQRY FUNC=STATS request directed to SCI. When the user exit is driven, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the SCI statistics header. When the CSLZQRY FUNC=STATS request is driven, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the SCI statistics header. The header is mapped by CSLSSTX.

Subsections:

- "SCI statistics header CSLSSTX" on page 621
- "SCI statistics record CSLSST1" on page 621
- "SCI statistics record CSLSST2" on page 622
- "SCI member statistics record CSLSST3" on page 623

SCI statistics header CSLSSTX

The following table lists the SCI Statistics Header CSLSSTX. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 281. SCI statistics header CSLSSTX

Field name	Offset	Length	Field usage	Description
SSTX_ID	X'00'	X'08'	Input	Eye catcher "CSLSSTX".
SSTX_LEN	X'08'	X'04'	Input	Length of header.
SSTX_PVER	X'0C'	X'04'	Input	Header version number (X'0000001').
SSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.
SSTX_STATOFF	X'14'	X'04'	Input	Offset to statistics area for first IMSplex. This is the offset from the beginning of CSLSSTX. The offset points to the CSLSST1 area.
SSTX_SST1OFF	X'18'	X'04'	Input	Offset to the SCI request statistics record for activity performed by SCI requests (mapped by macro CSLSST1). The offset is from the start of the statistics area for this IMSplex. Refer to the next table for a description of the SCI Request statistics record.
SSTX_SST2OFF	X'1C'	X'04'	Input	Offset to SCI IMSplex statistics record for activity performed by SCI for an IMSplex (mapped by macro CSLSST2). The offset is from the start of the statistics area for this IMSplex. Refer to Table 283 on page 622 for a description of the SCI IMSplex statistics record.
SSTX_SST3OFF	X'20'	X'04'	Input	Offset to first SCI member statistics record for SCI activity performed by each member in an IMSplex (mapped by the CSLSST3 macro). The offset is from the start of the statistics area for each IMSplex. Refer to Table 284 on page 623.
	X'24'	X'04'	Input	Reserved.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

SCI statistics record CSLSST1

CSLSST1 contains statistics that are related to requests that are processed by SCI. The following table lists the SCI Statistics Record CSLSST1. Included are the field name, the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 282. SCI statistics record CSLSST1

Field name	Offset	Length	Field usage	Description
SST1_ID	X'00'	X'08'	Input	Eye catcher "CSLSST1".
SST1_LEN	X'08'	X'04'	Input	Length of CSLSTT1 data.

Table 282. SCI statistics record CSLSST1 (continued)

Field name	Offset	Length	Field usage	Description
SST1_PVER	X'0C'	X'04'	Input	Statistics Version Number (X'00000001').
SST1_SCREG	X'10'	X'04'	Input	Number of local registrations.
SST1_RREG	X'14'	X'04'	Input	Number of remote registrations.
SST1_NREG	X'18'	X'04'	Input	Number of notify remote registrations.
SST1_SCRDY	X'1C'	X'04'	Input	Number of local readys.
SST1_RRDY	X'20'	X'04'	Input	Number of remote readys.
SST1_NRDY	X'24'	X'04'	Input	Number of notify remote readys.
SST1_SCQSC	X'28'	X'04'	Input	Number of local quiesces.
SST1_RQSC	X'2C'	X'04'	Input	Number of remote quiesces.
SST1_SCDRG	X'30'	X'04'	Input	Number of normal local deregistrations.
SST1_SCDRGA	X'34'	X'04'	Input	Number of abnormal local deregistrations.
SST1_RDRG	X'38'	X'04'	Input	Number of normal remote deregistrations.
SST1_RDRA	X'3C'	X'04'	Input	Number of abnormal remote deregistrations.
SST1_NABN	X'14'	X'04'	Input	Number of notify abends.
SST1_SCMI	X'40'	X'04'	Input	Number of member initializations.
	X'44'	X'04'	Input	Reserved.
	X'48'	X'04'	Input	Reserved.
	X'4C'	X'04'	Input	Reserved.
	X'50'	X'04'	Input	Reserved.
	X'54'	X'04'	Input	Reserved.
	X'58'	X'04'	Input	Reserved.
	X'5C'	X'04'	Input	Reserved.
	X'60'	X'04'	Input	Reserved.
	X'64'	X'04'	Input	Reserved.

SCI statistics record CSLSST2

CSLSST2 contains statistics that are related to an IMSplex, but not to a specific request. The following table lists the SCI Statistics Record CSLSST2. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 283. SCI statistics record CSLSST2

Field name	Offset	Length	Field usage	Description
SST2_ID	X'00'	X'08'	Input	Eye catcher "CSLSST2".
SST2_LEN	X'08'	X'04'	Input	Length of CSLSST2 data.
SST2_PVER	X'0C'	X'04'	Input	Statistics Version Number (X'00000001').

Table 283. SCI statistics record CSLSST2 (continued)

Field name	Offset	Length	Field usage	Description
SST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
SST2_SST3CNT	X'18'	X'04'	Input	Number of CSLSST3 records to follow.
SST2_MSGOGOOD	X'1C'	X'04'	Input	Number of successful IXCMSGO calls.
SST2_MSGOBFSH	X'20'	X'04'	Input	Number of IXCMSGO calls with buffer shortage.
SST2_MSGORSSH	X'24'	X'04'	Input	Number of IXCMSGO calls with other resource shortage.
	X'28'	X'04'	Input	Reserved.
	X'2C'	X'04'	Input	Reserved.
	X'30'	X'04'	Input	Reserved.
	X'34'	X'04'	Input	Reserved.
	X'38'	X'04'	Input	Reserved.
	X'3C'	X'04'	Input	Reserved.
	X'40'	X'04'	Input	Reserved.
	X'44'	X'04'	Input	Reserved.

SCI member statistics record CSLSST3

CSLSST3 contains statistics that are related to specific members of an IMSplex. There is one CSLSST3 entry for each registered IMSplex member when statistics are taken. The following table lists the SCI Statistics Record CSLSST3. Included are the offset value and length (both in hexadecimal), how the field is used, and a brief description of the field.

Table 284. SCI member statistics record CSLSST3

Field name	Offset	Length	Field usage	Description
SST3_ID	X'00'	X'08'	Input	Eye catcher "CSLSST3".
SST3_LEN	X'08'	X'04'	Input	Length of CSLSST3 data.
SST3_PVER	X'0C'	X'04'	Input	Statistics Version Number (X'00000001').
SST3_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
SST3_MBRNAME	X'18'	X'04'	Input	Member name.
SST3_MBRTYPE	X'20'	X'04'	Input	Member type.
SST3_RQSNTBYL	X'24'	X'04'	Input	Number of requests sent by this member to members on this system (local).
SST3_RQSNTBYR	X'28'	X'04'	Input	Number of requests sent by this member to members on remote systems.
SST3_RQSNTTO	X'2C'	X'04'	Input	Number of requests sent to this member by members on this system (local).
SST3_RQRCVBY	X'30'	X'04'	Input	Number of requests received by this member from all sources.

Table 284. SCI member statistics record CSLSST3 (continued)

Field name	Offset	Length	Field usage	Description
SST3_MGSNTBYL	X'34'	X'04'	Input	Number of messages sent by this member to members on this system (local).
SST3_MGSNTBYR	X'38'	X'04'	Input	Number of messages sent by this member to members on remote systems.
SST3_MBSNTBYM	X'3C'	X'04'	Input	Number of messages sent by this member to multiple members.
SST3_MGSNTTO	X'40'	X'04'	Input	Number of messages sent to this member by members on this system (local).
SST3_MGRCVBY	X'44'	X'04'	Input	Number of messages received by this member from all sources.
SST3_RQSTMOUT	X'48'	X'04'	Input	Number of requests sent to this member that timed out.
SST3_RQSLOST	X'4C'	X'04'	Input	Number of requests sent to this member that were lost due to an abend or lose system.
	X'50'	X'04'	Input	Reserved.
	X'54'	X'04'	Input	Reserved.
	X'58'	X'04'	Input	Reserved.
	X'5C'	X'04'	Input	Reserved.
	X'60'	X'04'	Input	Reserved.
	X'64'	X'04'	Input	Reserved.
	X'68'	X'04'	Input	Reserved.
	X'6C'	X'04'	Input	Reserved.

Related reference

"BPE Statistics user-supplied exit routine" on page 511
The BPE Statistics user-supplied exit routine is called at regular intervals during the life of a BPE address space, and a final time at normal address shutdown, to gather address-space related statistics.

Part 3. CQS client exit routines

CQS client exit routines allow a CQS client to monitor the CQS environment.

This topic contains Product-sensitive Programming Interface information.

CQS routines are written and supplied by a client (such as IMS). Each client must write its own exit routines tailored to the needs of that client product, to be supplied as part of the product. No sample CQS client exit routines are provided. The exit routines are given control in the client's address space in one of these two ways:

- For authorized clients (those running in supervisor state, key 0-7), the exits receive control in service request block (SRB) mode.
- For non-authorized clients (those running in problem state or non-key 0-7), the exits receive control as an interrupt request block (IRB) under the client task control block (TCB) that owns the cross memory resources for the address space (the TCB pointed to by ASCBXTCB).

Because each call to a client exit routine runs under its own SRB, the order in which the exits are driven is not guaranteed. It is possible for client exit routines to be driven out of order (different from the order from which CQS scheduled them). Your exit routines must be able to tolerate events that are received out of order. All client exit routine parameter lists contain an 8-byte time stamp in STCK format that is the time when CQS scheduled the SRB for the exit routine. This time stamp can be used to help determine the original order of events.

Related reference

“BPE-based CQS user-supplied exit routines” on page 547
Use BPE-based CQS user exit routines to customize and monitor your CQS environment.

Chapter 10. Client CQS Event exit routine

The CQS Event exit routine is driven when an event occurs in CQS that is related to CQS itself and might require some action to be taken by the client.

The client loads the exit routine and passes the exit routine address on the CQSREG request. This exit routine is driven in the client address space, either as an SRB (for authorized clients), or as an IRB (for non-authorized clients). The CQS Event exit routine is required.

The following CQS events drive the CQS Event exit routine:

- CQS initialization - client can reconnect to CQS
- CQS termination - abnormal termination

Subsections:

- "CQS restart entry parameter list" on page 628
- "CQS abnormal termination parameter list" on page 628
- "Client processing after CQS abnormal termination or restart" on page 629

Contents of registers on entry

Register
Contents

0
Length in bytes of the parameter list pointed to by R1.

1
Address of CQS Event Exit Parameter List (mapped by macro CQSCEVX).

13
Address of a standard 18-word save area, immediately followed by an 18-word work area that is available for the exit routine's use. The save area and the work area are not chained together. The save area or work area storage is not cleared on entry to the CQS Event exit routine.

14
Return address.

15
Entry point of exit routine.

Restriction: All addresses passed to the CQS Event Exit routine are valid only until the exit routine returns to its caller. These addresses should never be stored and used after the CQS Event exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses might have changed, or it might have been freed.

Contents of registers on exit

The CQS Event exit routine must preserve the contents of register 13; it does not need to preserve any other register's contents. Therefore, it is free to use the save area pointed to by register 13 for any calls to other services as needed (it can also use the 18-word area following the save area for additional save area or work area storage).

Register
Contents

13
The same value it had on entry to the CQS Event exit routine.

15
Return code

0
Always set this to zero.

CQS restart entry parameter list

The following table describes the CQS restart entry parameters for the Client CQS Event exit routine.

Table 285. Client CQS Event exit routine parameter list: CQS restart entry

Field name	Offset	Length	Description
CEVX_PVSN	X'00'	X'04'	Parameter list version number (X'00000001').
CEVX_EVENT	X'04'	X'04'	CQS event code **X'1'** CQS Initialization Event (CEVX_INIT).
CEVX_SCODE	X'08'	X'04'	CQS event subcode **X'1'** Client can re-register and reconnect to CQS (CEVX_RESTART).
CEVX_DATA	X'0C'	X'04'	Event exit routine client data that was passed to CQS on the CQSREG request.
CEVX_CQSID	X'10'	X'08'	CQS identifier.
CEVX_CQSVER	X'18'	X'04'	CQS version number.
CEVX_TSTMP	X'1C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).

CQS abnormal termination parameter list

The following table describes the CQS abnormal termination parameters for the Client CQS Event exit routine.

Table 286. Client CQS Event exit routine parameter list: CQS abnormal termination

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	CQS event code **2** CQS Termination Event.
X'08'	X'04'	CQS Event subcode **1** CQS abnormal termination entry. The CQS address space is terminating abnormally.
X'0C'	X'04'	Event exit routine client data that was passed to CQS on the CQSREG request.

Table 286. Client CQS Event exit routine parameter list: CQS abnormal termination (continued)

Offset	Length	Description
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'24'	X'04'	Abnormal Termination reason code. (CQS abend code)

Client processing after CQS abnormal termination or restart

If a client is registered with CQS and CQS terminates abnormally, the client's CQS Event exit routine is called with a CQS abnormal termination event. The client can choose to wait for CQS to be restarted, at which time the client's CQS Event exit routine is scheduled for a CQS restart event. When the CQS restart event is received, the client must perform the following steps before it can resume making CQS requests:

1. The client must reregister with CQS using the CQSREG macro. This step is necessary to reestablish the cross-memory connections between the client and CQS. Failure to reregister can result in an S0D6 abend when the next CQS request is issued.
2. The client must reconnect, using the CQSCONN macro, to any structures it was using prior to the CQS failure.
3. The client must resync indoubt UOWs with CQS, using the CQSRSYNC macro.
4. The client must register interest in queues, using the CQSINFRM request. If CQS terminated abnormally, it lost all previous client registration information.

Chapter 11. CQS Client Structure Event exit routine

The Client Structure Event exit routine is driven when an event occurs concerning a CQS-managed structure that might require some action to be taken by the client.

The client loads the exit routine and passes the address of the exit routine on the CQSCONN request. This exit routine is driven in the client address space, either as an SRB (for authorized clients), or as an IRB (for non-authorized clients). This exit routine is required, and applies both to resource and queue structures.

The following structure events drive the Client Structure Event exit routine:

- Resync UOW processing
  - When CQS Resync processing completes for an individual UOW, which had been deferred.
  - When CQS Resync processing occurs for the list of client UOWs that were not passed during the CQS Resync request.
  - **Important:** Resync UOW Processing only applies to queue structures.
- Checkpoint event
  - When structure checkpoint begin, end, or failure occurs.
  - **Important:** The Checkpoint event only applies to queue structures.
- Structure rebuild event
  - When structure copy (rebuild) begin, end, or failure occurs.
  - When structure recovery (rebuild) begin, end, or failure occurs.
  - When structure recovery lost UOWs occurs.
- Structure overflow event
  - When one or more queues move to the overflow structure.
  - When one or more queues move from the overflow structure. This event also indicates when the structure is no longer in overflow mode.
  - **Important:** The Structure Overflow event only applies to queue structures.
- Structure status change event
  - When the structure is available again after a loss.
  - When the structure fails. For resource structures only, failure means that CQS cannot allocate a new resource structure.
  - When CQS is able to repopulate (allocate) a new resource structure.
  - When CQS loses its connection to the structure.
  - When the log stream becomes available, making the structure available.

Subsections:

- “Structure status change parameter list for CQS Client Structure Event” on page 639

Contents of registers on entry

Register
Contents

0
Length in bytes of the parameter list pointed to by R1.

1
Address of Client Structure Event exit routine parameter list (mapped by macro CQSSEVX).

13
Address of a standard 18-word save area, immediately followed by an 18-word work area that is available for use by the exit routine. The save area and the work area are not chained together. The save area or work area storage is not cleared on entry to the Client Structure Event exit routine.

14
Return address.

15
Entry point of exit routine.

Restriction: All addresses that are passed to the Client Structure Event exit routine are valid only until the exit routine returns to its caller. These addresses should never be stored and used after the CQS Client Structure Event exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses might have changed, or it might have been freed.

Contents of registers on exit

The Client Structure Event exit routine must preserve the contents of R13; it does not need to preserve any other register contents. Therefore, it is free to use the save area pointed to by R13 for any calls to other services as needed. The exit routine can also use the 18-word area following the save area for additional save area or work area storage.

Register
Contents

13
The same value it had on entry to the Client Structure Event exit routine.

15
Return code

X'00'
Always set this to zero.

Deferred resync complete parameter list for CQS Client Structure Event

The following table describes the deferred resync complete parameters for the Client Structure Event exit routine.

Table 287. Client Structure Event exit routine parameter list: deferred resync complete

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	Structure event code **1** Resync UOW event.

Table 287. Client Structure Event exit routine parameter list: deferred resync complete (continued)

Offset	Length	Description
X'08'	X'04'	Structure event subcode **1** Deferred resync complete.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure Name.
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'20'	Unit of work (UOW) identifier.
X'54'	X'10'	Queue name.
X'64'	X'10'	Deferred resync token. This is the Put token that is used for `Put Forget` processing.
X'74'	X'02'	CQS UOW state **X'0010'** Put Insync Client status is Put Complete. CQS status is Put Complete. CQS knows about the UOW and all data objects for the UOW are out on the coupling facility. A PUT token is returned for the UOW. The client should use the PUT token to issue the CQSPUT FUNC=FORGET request. **X'00F2'** Unknown Client status is Put Complete. CQS has no knowledge of the UOW. If the client believes the UOW is in Put Complete status, the client must determine whether to reissue the CQSPUT requests.
X'76'	X'02'	Reserved.

CQS resync parameter list

The following table describes the CQS initiated resync parameters for the Client Structure Event exit routine.

Table 288. Client Structure Event Routine exit parameter list: CQS initiated resync

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	Structure event code **1** Resync UOW event.

Table 288. Client Structure Event Routine exit parameter list: CQS initiated resync (continued)

Offset	Length	Description
X'08'	X'04'	Structure event subcode **2** CQS Initiated resync processing.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure name.
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'04'	Number of unit of work (UOW) list entries.
X'38'	X'04'	Length of each UOW list entry.
X'3C'	X'04'	Offset into parameter list of start of UOW list. The parameter list is one contiguous piece of storage, including the UOW list.

CQS resync UOW entry

The following table describes the CQS resync UOW entry parameters for the Client Structure Event exit routine.

Table 289. CQS resync UOW entry parameters

Offset	Length	Description
X'00'	X'20'	Unit of work (UOW) identifier.
X'20'	X'10'	Queue name.
X'30'	X'10'	Resync token. • If the CQS UOW status is locked, this field contains a lock token. This lock token is to be used on subsequent requests, such as CQSREAD and CQSUNLCK to process the locked data object. • If the CQS UOW status is `COLD QUEUE`, this field contains a cold queue token. This cold queue token is to be used along with the UOW on a CQSRECVR request to recover the data object on the cold queue.

Table 289. CQS resync UOW entry parameters (continued)

Offset	Length	Description
X'40'	X'02'	CQS UOW status **X'00F1'** Locked. This data object is locked. A lock token is passed back to the client in the Resync token field. This token field is required on subsequent requests to process the locked data object. **X'00F3'** Cold Queue: CQS-Client Cold Start. This data object is on the cold queue because of either a CQS cold start or client cold start. A cold queue token is passed back to the client in the Resync token field. This token field is required on a subsequent CQSRECVR request to process the data object on the cold queue. **X'00F4'** Cold Queue: Unknown. This data object is on the cold queue. CQS warm started after a structure rebuild from the log took place and the object was found locked by CQS. A cold queue token is passed back to the client in the Resync token field. This token field is required on a subsequent CQSRECVR request to process the data object on the cold queue. This status is returned only for structures defined with BATCHDEL=YES coded (or defaulted to) on their STRUCTURE definition in the CQSSL*xxx* PROCLIB member.
X'42'	X'02'	Reserved.

Checkpoint parameter list for CQS Client Structure Event

The following table describes the checkpoint parameters for the Client Structure Event exit routine.

Table 290. Client Structure Event exit routine parameter list: checkpoint

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	Structure event code **2** Checkpoint event.
X'08'	X'04'	Structure event subcode **1** Structure checkpoint begin. **2** Structure checkpoint end. **3** Structure checkpoint failure.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure name.

Table 290. Client Structure Event exit routine parameter list: checkpoint (continued)

Offset	Length	Description
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'08'	CQS identifier of the master CQS performing the checkpoint process.
X'3C'	X'01'	Flag byte. **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same.
X'3D'	X'03'	Reserved.

Structure rebuild parameter list for CQS Client Structure Event

The following table describes the structure rebuild parameters for the Client Structure Event exit routine.

Table 291. Client Structure Event exit routine parameter list: structure rebuild

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	Structure event code **3** Structure rebuild event.
X'08'	X'04'	Structure event subcode **1** Structure rebuild begin. **2** Structure rebuild (copy) end. **3** Structure rebuild (copy) failure. **4** Structure rebuild failure. **5** Structure rebuild (recovery) end. **6** Structure rebuild (recovery) failure.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure name.
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'08'	CQS identifier of the master CQS performing the rebuild process.

Table 291. Client Structure Event exit routine parameter list: structure rebuild (continued)

Offset	Length	Description
X'3C'	X'01'	Flag byte. **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same.
X'3D'	X'03'	Reserved.

Structure rebuild lost UOWs parameter list for CQS Client Structure Event

The following table describes the structure rebuild lost UOW parameters for the Client Structure Event exit routine. These UOWs are nonrecoverable and were lost by the last structure recovery. Some of the UOWs in the list might belong to other clients if the structure recovery occurred while CQS was down.

Table 292. Client Structure Event exit routine parameter list: structure rebuild lost UOWs

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	Structure event code **3** Structure rebuild event.
X'08'	X'04'	Structure event subcode **7** Structure recovery lost UOWs. **Important:** This subcode applies only to queue structures.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure name.
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'08'	CQS identifier of the master CQS performing the rebuild process.
X'3C'	X'01'	Flag byte. **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same.
X'3D'	X'03'	Reserved.
X'40'	X'04'	Number of Lost UOW list entries.
X'44'	X'04'	Length of each Lost UOW list entry.
X'48'	X'04'	Offset into parameter list of start of Lost UOW list. The parameter list is one contiguous piece of storage, including the Lost UOW list.

Rebuild lost UOW entry for CQS Client Structure Event

The following table describes the CQS rebuild lost UOW entry parameters for the Client Structure Event exit routine.

Table 293. CQS rebuild lost UOW entry parameters

Offset	Length	Description
X'00'	X'20'	Unit of work (UOW) identifier.
X'20'	X'10'	Client queue name.
X'30'	X'01'	Lost UOW status. **X'80'** Lost UOW was on client queue. **X'40'** Lost UOW was locked. **X'20'** Lost UOW was on COLDQ. **X'10'** Lost UOW was on CQS private queue.
X'31'	X'03'	Reserved.

Structure overflow parameter list for CQS Client Structure Event

The following table describes the structure overflow parameters for the Client Structure Event exit routine.

Table 294. Client Structure Event exit routine parameter list: structure overflow

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000001').
X'04'	X'04'	Structure event code **4** Structure overflow event.
X'08'	X'04'	Structure event subcode **1** Move queues to overflow. One or more queues was selected as candidates to be moved to the overflow structure and was approved by the Queue Overflow user exit routine. **2** Move queues from overflow. One or more queues moved from the overflow structure back to the primary structure, because the queues were drained on the overflow structure. New work for these queues is placed on the primary structure.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure name.

Table 294. Client Structure Event exit routine parameter list: structure overflow (continued)

Offset	Length	Description
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'08'	CQS identifier of the master CQS performing the overflow process.
X'3C'	X'01'	Flag byte. **X'80'** This CQS is the master of the process. The CQS identifier and master CQS identifier are the same. **X'40'** The structure is no longer in overflow mode. This value applies only to subcode 2.
X'3D'	X'03'	Reserved.
X'40'	X'04'	Number of queue name entries in the list.
X'44'	X'04'	Length of each queue name list entry.
X'48'	X'04'	Offset into parameter list of start of queue name list. Each queue name list entry contains the 16-byte queue name of a queue that is being moved to the overflow structure. The parameter list is one contiguous piece of storage, including the queue name list.

Structure status change parameter list for CQS Client Structure Event

The following table describes the structure status change parameters for the Client Structure Event exit routine.

Table 295. Client Structure Event exit routine parameter list: structure status change

Offset	Length	Description
X'00'	X'04'	Parameter list version number (00000002).
X'04'	X'04'	Structure event code. **5** Structure status change event.

Table 295. Client Structure Event exit routine parameter list: structure status change (continued)

Offset	Length	Description
X'08'	X'04'	Structure event subcode **1** Structure available again after a loss. **2** The structure failed. **3** CQS lost its connection to the structure (STXLCONN). **4** The log stream is becoming available, making the structure available (STXAVLOG). **Important:** This subcode applies only to queue structures. **5** The log stream is becoming unavailable, making the structure unavailable (STXFLOG). **Important:** This subcode applies only to queue structures. **6** Structure repopulation required due to structure failure.
X'0C'	X'04'	Structure Event exit routine client data that was passed to CQS on the CQSCONN request.
X'10'	X'08'	CQS identifier.
X'18'	X'04'	CQS version number.
X'1C'	X'10'	Structure Name.
X'2C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'34'	X'01'	Structure type **1** Queue structure **2** Resource structure
X'38'	X'18'	Not used.
X'50'	X'08'	Structure version of new structure that requires repopulation, because old structure failed.

Chapter 12. CQS Client Structure Inform exit routine

The CQS Client Structure Inform exit routine is scheduled when work is placed on a queue for which the client has registered interest with a CQSINFRM request and when a CQSINFRM request is issued specifying that the exit routine be driven if there is work on the queue.

The exit routine is also scheduled whenever a queue goes from an empty to non-empty state (when the first data object for a queue is written to the structure). If additional data objects are added to the queue, the inform exit routine, which has already been run once, is not notified again while there are still data objects on the queue.

The client loads the exit routine and passes the address of the exit routine on the CQSCONN request. This exit routine is driven in the client address space, either as an SRB (for authorized clients), or as an IRB (for non-authorized clients).

Restriction: This exit routine does not apply to resource structures.

Important: This exit routine is optional; however, if it is not supplied, the client is not notified when work is placed on the queues.

Contents of registers on entry

Register
Contents

0
Length in bytes of the parameter list pointed to by register 1.

1
Address of CQS Structure Inform Exit Parameter List (mapped by macro CQSINFX).

13
Address of a standard 18-word save area, immediately followed by an 18-word work area available for use by the exit routine. The save area and the work area are not chained together. The save area or work area storage is not cleared on entry to the Structure Inform Exit routine.

14
Return address.

15
Entry point of exit routine.

Restriction: All addresses that are passed to the CQS Structure Inform exit routine are valid only until the exit routine returns to its caller. These addresses should never be stored and used after the CQS Structure Inform exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses might have changed, or it might have been freed.

Contents of registers on exit

The CQS Structure Inform exit routine must preserve the contents of register 13 and it does not need to preserve any other register's contents. Therefore, it is free to use the save area pointed to by register 13 for any calls to other services as needed. It might also use the 18-word area following the save area for additional save area or work area storage.

Register
Contents

13
Same value as it had on entry to the CQS Structure Inform exit routine.

15
Return code

0
Always set this to zero.

Structure inform parameter list for CQS Client Structure Inform

The following table describes the parameters for the Client Structure Inform exit routine.

Table 296. Client Structure Inform exit routine parameter list

Offset	Length	Description
X'00'	X'04'	Parameter list version number (X'00000002').
X'04'	X'04'	Structure Inform exit routine client data that was passed to CQS on the CQSCONN request.
X'08'	X'08'	CQS identifier.
X'10'	X'04'	CQS version number.
X'14'	X'10'	Structure name.
X'24'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
X'2C'	X'04'	Number of queue name entries in the list.
X'30'	X'04'	Length of each queue name list entry.
X'34'	X'04'	Offset into parameter list of start of queue name list. Each queue name entry in the list contains the 16-byte queue name for which a message has been queued. The parameter list is one contiguous piece of storage, including the queue name list.
X'38'	X'08'	Time stamp of the time that the CQS list transition exit was driven in STCK format. This field only exists when the parameter list version number (at offset X'00') is X'00000002' or higher.

Part 4. CSL SCI IMSplex member exit routines

This topic describes the exits that SCI can drive in the address space of a registered IMSplex member.

These exit routines allow an IMSplex member to:

- Monitor what address spaces are active members of the IMSplex.
- Receive messages and requests from other members of the same IMSplex.

SCI member exits are written and supplied by an IMSplex member (such as the IMS control region). Each member must write its own exit routines tailored to the needs of that member product, to be supplied as part of the product. No sample SCI exit routines are provided. The exit routines are given control in the member's address space in one of two ways:

- For authorized members (those running in supervisor state, key 0-7), the exits receive control in SRB mode.
- For non-authorized members (those running in problem state or non-key 0-7), the exits receive control as an IRB under the member TCB associated with the SCI registration.

Because each call to a member exit routine runs under its own SRB, the order in which the exits are driven is not guaranteed. It is possible for member exit routines to be driven out of order (different from the order in which SCI scheduled them). Your exit routines must be able to tolerate events that are received out of order. All member exit routine parameter lists contain an 8-byte time stamp in STCK format, which is the time when SCI scheduled the SRB for the exit routine. This time stamp can be used to help determine the original order of events.

Related reference

“BPE-based CSL SCI user exit routines” on page 616
SCI user exits allow you to customize and monitor the SCI environment. They are written and supplied by the user. No sample exits are provided.

Chapter 13. CSL SCI Input exit routine

The SCI Input exit routine is called whenever there is a message or a request for the IMSplex member.

The IMSplex member loads the exit routine and passes the exit routine address on the CSLSCREG request. The exit is driven in the member's address space, either as an SRB (for authorized members) or as an IRB (for non-authorized members).

Subsections:

Contents of registers on entry

Register
Contents

0
Length in bytes of the parameter list pointed to by R1.

1
Address of SCI Input exit parameter list (mapped by macro CSLSINXP).

13
Address of 2 pre-chained save areas. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the exit.

14
Return address.

15
Entry point of exit routine.

Restriction: All addresses passed to the SCI Input Exit routine are valid only until the exit routine returns to its caller (with the exception of the member parameter list address). These addresses should never be stored and used after the SCI Input Exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses can change or be reassigned by IMS after the exit returns. The member parameter list address is the exception to this restriction. It is available until the storage is released by issuing the CSLSCBFR FUNC=RELEASE request (for messages), or the CSLSCRQR FUNC=RETURN request (for requests).

Contents of registers on exit

The SCI Input exit routine must preserve the contents of R13; it does not need to preserve any other register's contents. Therefore, it can use the save areas pointed to by R13 for any calls to other services as needed.

Register
Contents

13
The same value it had on entry to the SCI Input exit routine.

15
Return code

0
The message or request was successfully received.

4
The message or request was not received because the destination member did not understand the function. If the input data is for a request, SCI sends a response with return code=SRC_PARM (parameter error) and reason code=SRSN_FUNCTION (invalid function). SCI releases storage for

any parameters that SCI allocated. If the input data is for a message, SCI releases the storage that contains the message.

8

The message or request was not received because of an internal error. If the input data is for a request, SCI sends a response with return code=SRC_SYSTEM (system error) and reason code=SRSN_INTERNAL (internal error). SCI releases the storage for any parameters that SCI allocated. If the input data is for a message, SCI releases the storage that contains the message.

CSL SCI input exit parameter list

The following table describes the entry parameters for the parameter list header of the Client SCI Input exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 297. Client SCI input exit routine parameter list: parameter list header

Field name	Offset	Length	Description
INXP_PVER	X'00'	X'04'	Parameter list version number (X'00000001').
INXP_PLEN	X'04'	X'04'	Total length of parameter list.
INXP_SCIVSN	X'08'	X'04'	Version of SCI on the system from which this message or request originated.
INXP_EXITPARM	X'0C'	X'08'	Input exit routine member data that was passed to SCI on the CSLSCREG request with the INPUTPARM parameter. If no data was passed on the CSLSCREG request, this field contains zeros.
INXP_PLEXNAME	X'14'	X'08'	IMSplex name.
INXP_TIMESTMP	X'1C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
INXP_DATAOFF	X'24'	X'04'	Offset of Message Data Section from the start of the parameter list header.
INXP_SRCOFF	X'28'	X'04'	Offset of Source Member Data Section from the start of the parameter list header

The following table describes the entry parameters for the message data of the Client SCI Input exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 298. Client SCI input exit routine parameter list: message data

Field name	Offset	Length	Description
INXP_FUNC	X'00'	X'04'	Function code.
INXP_SFUNC	X'04'	X'04'	Subfunction code.

Table 298. Client SCI input exit routine parameter list: message data (continued)

Field name	Offset	Length	Description
INXP_DATAFL1	X'08'	X'01'	Data Flag. **X'80'** INXP_RQST This bit indicates that the input data is a request. When the receiver of the request has completed processing the request, it must be returned using the CSLSCRQR request. If the bit is not set, the input data is a message. When the receiver of the message has completed processing the message, it should return the storage to SCI using the CSLSCBFR request. **X'40'** INXP_FTYPSND When this bit is on, the function code in INXP_FUNC is defined by the sender. When this bit is off, the function code is defined by the destination.
	X'09'	X'03'	Reserved.
INXP_MBRPLCNT	X'0C'	X'04'	The number of parameters (pairs of lengths and addresses) passed in the member parameter list.
INXP_MBRPLPTR	X'10'	X'04'	The address of the member parameter list.
	X'14'	X'04'	Reserved.
INXP_RQSTTKN	X'18'	X'08'	Request token. This field is valid only if bit INXP_RQST is set (indicating that this is a request). The request token is used to return the request to the sender when issuing the CSLSCRQR request. If INXP_RQST is not set (indicating this is a message), this field is unused.
	X'20'	X'04'	Reserved.
	X'24'	X'04'	Reserved.

The following table describes the entry parameters for the input source data of the Client SCI Input exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 299. Client SCI input exit routine parameter list: input source data

Field name	Offset	Length	Description
INXP_SCITKN	X'00'	X'10'	The SCITOKEN of the IMSplex member that is the source of this data.
INXP_MBRNAME	X'10'	X'08'	The name of the IMSplex member that is the source of this data.
INXP_MBRVSN	X'18'	X'04'	The version of the IMSplex member that is the source of this data. If the source IMSplex member did not pass a MBRVSN on the CSLSCREG request, this field is set to zeros.

Table 299. Client SCI input exit routine parameter list: input source data (continued)

Field name	Offset	Length	Description
INXP_TYPE	X'1C'	X'02'	The IMSplex member type of the IMSplex member that is the source of this data.
	X'1E'	X'02'	Reserved.
INXP_SUBTYPE	X'20'	X'08'	The subtype of the IMSplex member that is the source of this data. If the source IMSplex member did not pass a SUBTYPE on the CSLSCREG request, this field is set to zeros.
INXP_JOBNAME	X'28'	X'08'	The jobname of the IMSplex member that is the source of this data.
INXP_USERID	X'30'	X'08'	The user ID of the IMSplex member that is the source of this data.
INXP_SRCFL1	X'38'	X'01'	Source Flag **X'80'** This bit indicates that the member that sent this data is authorized.
	X'39'	X'03'	Reserved.
	X'3C'	X'04'	Reserved.
	X'40'	X'04'	Reserved.
	X'44'	X'04'	Reserved.

Chapter 14. CSL SCI Notify Client exit routine

The SCI Notify exit routine is driven whenever there is a change in the SCI status of an IMSplex member. This allows a member to keep track of the status of other members in the IMSplex.

The IMSplex member loads the exit routine and passes the exit routine address on the CSLSCREG request. The exit is driven in the member's address space, either as an SRB (for authorized members) or as an IRB (for non-authorized members).

The exit is driven whenever an IMSplex member:

- Completes a successful CSLSCREG FUNC=REGISTER
- Completes a successful CSLSCRDY FUNC=READY
- Completes a successful CSLSCQSC FUNC=QUIESCE
- Completes a successful CSLSCDRG FUNC=DEREGISTER
- Terminates without issuing a CSLSCDRG FUNC=DEREGISTER request.
- Is not reachable because the local SCI is not active.

Note that some fields are not available in the Notify exit parameter list when the exit is driven for CSLSCDRG-related events (normal and abnormal termination) and when an IMSplex member is not reachable.

If the local SCI is the IMSplex member for which the Notify exit is being driven, the NXFP_LOCALSCI (X'40') bit in the NXFP_FLAG1 is set. When an SCI terminates, processing on the z/OS image for the IMSplex that was managed by the inactive SCI is limited until the SCI restarts:

- No messages or requests can be sent or received by any local IMSplex member.
- The SCI Notify exit cannot be driven for local IMSplex members for the following events:
  - CSLSCREG FUNC=REGISTER
  - CSLSCRDY FUNC=READY
  - CSLSCQSC FUNC=QUIESCE
  - CSLSCDRG FUNC=DEREGISTER (non-authorized member)
  - Termination without CSLSCDRG FUNC=DEREGISTER (non-authorized member)

  The Notify exit continues to be driven for normal and abnormal deregistrations for authorized members.
- No new members can join the IMSplex on the z/OS image.
- No SCI requests can be processed by local IMSplex members (for example, CSLSCQRY and CSLSCDRG requests).

When SCI restarts on the z/OS image, SCI re-registers each IMSplex member that is still active. The SCITOKEN for each IMSplex member is still valid. The Notify exit routine for each local member is driven for the following events:

- Registration for the local SCI
- Registration and Ready (if appropriate) for the local IMSplex members
- Ready for the local SCI
- Registration and Ready (if appropriate) for IMSplex members that are not local

Events for members that are not local can be scheduled before the Ready for the local SCI; however, events for local members are all scheduled before the SCI Ready event is scheduled. Local IMSplex members should not use SCI services until they have received the Ready event for the local SCI.

Notes:

- Since these events are sent using an SRB or an IRB, they might not be received by the member in their logical order. This is especially true with non-authorized members because the exit is called using an

IRB. If the next event occurs before the exit has been called by the IRB, the new IRB will interrupt the previous IRB, and the notification for the newer event will be received first. For example, the abnormal deregistration event could be received before the NOT-REACHABLE event. Since the abnormal deregistration event should occur later than the NOT-REACHABLE event, your program should be able to handle a NOT-REACHABLE event for a member that has already been abnormally deregistered.

- In rare cases, authorized members might receive two deregistration notifications for an authorized member. This is because SCI has a secondary path for authorized member deregistration that might be used when XCF notifications are delayed for some reason. If the delayed XCF notifications occur after the secondary path is used, a second notification might be sent.

Subsection:

- "CSL SCI notify exit parameter list" on page 650

Contents of registers on entry

Register
Contents

0
Length in bytes of the parameter list pointed to by R1.

1
Address of SCI Notify exit parameter list (mapped by macro CSLSNFXP).

13
Address of 2 prechained save areas. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the exit.

14
Return address.

15
Entry point of exit routine.

Restriction: All addresses passed to the SCI Notify exit routine are valid only until the exit routine returns to its caller. These addresses should never be stored and used after the SCI Notify exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses can be changed or reassigned by IMS after the exit returns.

Contents of registers on exit

The SCI Notify exit routine must preserve the contents of register 13; it does not need to preserve any other register's contents. Therefore, it is free to use the save areas pointed to by register 13 for any calls to other services as needed.

Register
Contents

13
The same value it had on entry to the SCI Notify exit routine.

15
Return code

0
Always set this to zero.

CSL SCI notify exit parameter list

The following table describes the parameter list header of the SCI Notify Client exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 300. SCI notify client exit routine parameter list header

Field name	Offset	Length	Description
NFXP_PVER	X'00'	X'04'	Parameter list version number (X'00000001').
NFXP_PLEN	X'04'	X'04'	Total length of parameter list.
NFXP_EXITPARM	X'08'	X'08'	Notify exit routine member data that was passed to SCI on the CSLSCREG request with the NOTIFYPARM parameter. If no data was passed on the CSLSCREG request, this field contains zeros.
NFXP_PLEXNAME	X'10'	X'08'	IMSplex name.
NFXP_SCIVSN	X'18'	X'04'	SCI Version
NFXP_TIMESTMP	X'1C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
NFXP_SUBJOFF	X'24'	X'04'	Offset of Subject Data Section.
	X'28'	X'04'	Reserved.

The following table describes the subject data of the SCI Notify Client exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 301. SCI notify client exit routine parameter list - subject data

Field name	Offset	Length	Description
NFXP_SCITKN	X'00'	X'10'	The SCITOKEN of the member that is the subject of this event.
NFXP_EVENT	X'10'	X'02'	The event that initiated this notification. **1** CSLSCREG FUNC=REGISTER **2** CSLSCRDY FUNC=READY **3** CSLSCQSC FUNC=QUIESCE **4** CSLSCDRG FUNC=DEREGISTER **5** Termination without CSLSCDRG FUNC=DEREGISTER **6** The member cannot be reached because the local SCI is not active.
NFXP_FLAG1	X'12'	X'01'	The event that initiated this notification. **X'80'** This bit indicates that the subject of this event is authorized. **X'40'** This bit indicates that the subject of this event is the local SCI.
	X'13'	X'01'	Reserved.

Table 301. SCI notify client exit routine parameter list - subject data (continued)

Field name	Offset	Length	Description
NFXP_MBRNAME	X'20'	X'08'	The Name of the IMSplex member that is the subject of this event.
NFXP_MBRVSN	X'28'	X'04'	The Version of the IMSplex member that is the subject of this event. If the subject IMSplex member did not pass a MBRVSN on the CSLSCREG request, this field is set to zeros. This data is not filled in for: • NFXP_EVENT= 4 (normal termination) • NXFP_EVENT= 5 (abnormal termination) • NXFP_EVENT=6 (not reachable)
NFXP_TYPE	X'14'	X'02'	The IMSplex member Type of the IMSplex member that is the subject of this event.
	X'16'	X'02'	Reserved.
NFXP_SUBTYPE	X'18'	X'08'	The Subtype of the IMSplex member that is the subject of this event. If the subject IMSplex member did not pass a SUBTYPE on the CSLSCREG request, this field is set to zeros.
NFXP_JOBNAME	X'18'	X'08'	The Jobname of the IMSplex member that is the subject of this event. This data is not filled in for: • NFXP_EVENT= 4 (normal termination) • NXFP_EVENT= 5 (abnormal termination) • NXFP_EVENT=6 (not reachable)

Part 5. IMS Connect exit routines

Code and modify IMS Connect exit routines to manage the messages to and from the various types of IMS Connect TCP/IP clients or provide general functionality, such as security and routing.

Chapter 15. IMS Connect 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 component
- Rerouting messages
- Checking security for input messages
- Returning user-defined messages in response to certain user-defined criteria

For security checking, the IMS Connect user message exit routines allow you to call IMSLSECX, the security message exit routine, issue the RACF function in these user message exit routines, or use the IMS Connect user RACF function.

This topic contains Product-Sensitive Programming Interface and Associated Guidance Information.

Note: Do not issue any MVS calls in the user message exit that result in an `MVS WAIT` because the `MVS WAIT` will halt all work on the port. If you modify the user message exit routine 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 exit routines cannot be modified to free any storage passed to the exit routine, and IMS Connect will not free any storage obtained by the user message exit routine when the exit routine 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 routine without causing failures.

User message exit routines HWSSMPL0 and HWSSMPL1

The user message exit routines HWSSMPL0 and HWSSMPL1 support IMS Connect client applications that are provided by your installation or another third party. The HWSSMPL0 and HWSSMPL1 exit routines manage the translation of message headers on input and output messages, and provide you with a point of control to modify, route, and check security for messages from and to the client.

The HWSSMPL0 and HWSSMPL1 exit routines and the related macros are shipped with IMS both as source code and as load modules. If you do not need to modify the exit routines, you can use the load modules. If you use the source code, you must assemble and bind the source code after you modify it for your needs.

If you need customized exit routines to support your user-provided IMS Connect clients, you can modify the source of either HWSSMPL0 or HWSSMPL1. You can also create new exit routines by modifying the source and renaming the modified exit routine. You can customize exit routines for specific clients without having to combine the logic for numerous clients in a single exit routine. Each different exit routine is identified by a unique name in the IRM_ID field of input messages.

The HWSSMPL0 and HWSSMPL1 user message exit routines provide the following functions:

- Perform data translation of ASCII to EBCDIC for input messages.
- Perform data translation of EBCDIC to ASCII for output messages.
- Support for Unicode transaction codes and application data. IMS Connect only translates the transaction code. Unicode application data is passed to the IMS host application without being translated.
- Build the IMS Connect message structure (BPE and OTMA headers).
- If IMSLSECX (the security message exit) is bound with either of these message exits, then the security message exit is called.

- Optionally, return a user-defined message with reason and return codes back to a client application as a reply to an input message without terminating the persistent socket connection. After sending the message, IMS Connect will terminate or keep a socket connection open depending on the return code set in the EXPREA_RETCODE parameter pointed to by register 1 at the READ subroutine exit.

  Important: IMS Connect does not support this function for Local Option connections. Additionally, the message length must be from a minimum of 1 to the maximum of 128 characters. If the specified message length is greater than 128, the message will be truncated to 128.
- By default, set the commit mode processing to 1 (send then commit) on input messages.
- By default, set the synchronization processing to NONE (SYNC LEVEL = NONE) on input messages, so that client is not required to return an acknowledgment.
- Set RACF options or passwords.
- If no Client ID is passed to the exit, the message exit generates the Client ID.
- Analyze the following option specifications in the headers of messages:
  - COMMIT MODE to override default.
  - SYNC LEVEL to override default.
  - LTERM to override logical terminal names
  - MFS MOD name.
  - ACK/NAK/DEALLOCATE.
  - RACF options.
- Optionally, issue a DFS2082 message for both RESPONSE and NONRESPONSE mode CM0 transactions when the IMS application does not reply to the IOPCB or complete a message switch to another transaction.

By default, the COMMIT mode is set to 1, and the SYNC LEVEL is set to NONE. These values can be overridden by supplying the COMMIT mode, the sync level, or both in the IRM_FLG2 and IRM_FLG3 fields of the message prefix received from the client. Alternatively, you can change the exit to set a different default values for COMMIT mode and SYNC level.

If the input from the client is in ASCII, the IMS Connect HWSSMPL0 and HWSSMPL1 exits translate ASCII to EBCDIC and build the required message structure containing the OTMA headers for messages received *from* the client. This exit performs the translation from EBCDIC to ASCII and removes the OTMA headers for messages being transmitted *to* the client. You can also modify the translation function of the exit routines to suit the needs of your client application.

These user exits call the user-provided security exit if one is defined to this exit and passes a parameter list in register 1.

Related concepts

Ping support for IMS Connect (Communications and Connections)

User-defined messages (Communications and Connections)

Related tasks

Changing RACF passwords by using client messages (Communications and Connections)

Related reference

HWSSMPL0 and HWSSMPL1 security actions (Communications and Connections)

IRM structures for IMS Connect client messages (Communications and Connections)

Input message from client and passed to message exit (Communications and Connections)

Input message returned from message exit (Communications and Connections)

Output message passed to message exit (Communications and Connections)

Output message from message exit to client (Communications and Connections)

HWSSMPL0 sample JCL

The following sample JCL is for the HWSSMPL0 user message exit.

```
//HWSSMPL JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//SMPL01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSSMPL0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMS.SDFSSMPL(HWSSMPL0),DISP=SHR
//SMPL02 EXEC PGM=IEWL,COND=(0,NE),
//        PARM='SIZE=(180K,28K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSSMPL0)
 ENTRY   HWSSMPL0
 MODE RMODE(24),AMODE(31)
 NAME HWSSMPL0(R)
//
```

HWSSMPL1 sample JCL

The following sample JCL is for the HWSSMPL1 user message exit routine.

```
//HWSSMPL JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//SMPL01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSSMPL1)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMS.SDFSSMPL(HWSSMPL1),DISP=SHR
//SMPL02 EXEC PGM=IEWL,
//        PARM='SIZE=(180K,28K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSSMPL1)
 ENTRY   HWSSMPL1
 MODE RMODE(24),AMODE(31)
 NAME HWSSMPL1(R)
//
```

IMS TM Resource Adapter user message exit routine (HWSJAVA0)

Use the IMS TM Resource Adapter user message exit routine (HWSJAVA0) to edit messages and perform custom security checking in support of the IMS Connect client, IMS TM Resource Adapter.

The HWSJAVA0 exit routine and its related macros are provided as both load modules and source code. Use the load module if you do not want to alter the way HWSJAVA0 operates. Edit the source code if you

want to use a modified version of the exit in your installation. After modifying the source code, you must assemble and link edit it to replace the pre-assembled load module in your system.

The HWSJAVA0 exit routine is bound into the IMS.SDFSRESL data set. This exit does not perform a translation or build to the OTMA headers. Both the translation and insertion or deletion of the OTMA header is done by IMS TM Resource Adapter.

If the OTMA header of a message from IMS TM Resource Adapter contains network security credentials and the HWSOMPFX macro is used, the HWSJAVA0 exit routine specifies both the **DSECT=** and the **NETSEC_OPT=**YES options for the HWSOMPFX macro. The **DSECT=** and the **NETSEC_OPT=**YES options cause the following behaviors:

- An individual DSECT is generated for each section of the OTMA message header.
- The HWSECDNDS DSECT, or the HWSECARDS DSECT, or both, is generated to map network security information.
- The HWSOMPFX DSECT is not generated.

By default, the HWSJAVA0 exit routine sets the COMMIT mode to 1, and the SYNC level to NONE. However, IMS TM Resource Adapter can override these settings.

The HWSJAVA0 exit routine can return user-defined messages and return and reason codes when user-defined criteria are met. The HWSJAVA0 exit routine can also request that IMS Connect keep the socket connection open after returning a user-defined message depending on the return code set in the EXPREA_RETCODE parameter pointed to by register 1 at the READ subroutine exit.

Important: IMS Connect does not support user-defined messages for Local Option connections. Additionally, the message length must be from a minimum of 1 to the maximum of 128 characters. If the specified message length is greater than 128, the message will be truncated to 128.

This user exit also calls the user-provided security exit (IMSLSECX) if one is defined to this exit and passes a parameter list in register 1.

Related concepts

Ping support for IMS Connect (Communications and Connections)

User-defined messages (Communications and Connections)

Related tasks

Changing RACF passwords by using client messages (Communications and Connections)

Related reference

“z/OS TCP/IP IMS Listener security exit (IMSLSECX)” on page 688

If any IMS Connect user message exit performs security checking, you must provide a security exit routine or use the z/OS TCP/IP IMS Listener security exit routine (IMSLSECX).

Message structures (Communications and Connections)

HWSJAVA0 sample JCL

The following sample JCL is for the HWSJAVA0 user message exit routine.

```
//HWSJAVA JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//JAVA01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSJAVA0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMS.SDFSSMPL(HWSJAVA0),DISP=SHR
//JAVA02 EXEC PGM=IEWL,
//        PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
```

```
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSJAVA0)
 ENTRY   HWSJAVA0
 NAME HWSJAVA0(R)
//
```

SOAP Gateway exit routine (HWSSOAP1)

The IMS Connect SOAP Gateway exit routine (HWSSOAP1) is required by IMS Connect if you use the IMS Enterprise Suite SOAP Gateway.

The HWSSOAP1 exit routine is shipped with IMS Connect and link-edited into the IMS.SDFSRESL data set. The source code for HWSSOAP1 is not shipped and cannot be modified or replaced.

The functionality of the HWSSOAP1 exit routine is similar to the HWSSMPL1 sample exit routine, except that HWSSOAP1 provides additional functionality to support the XML message conversion function that IMS Connect provides for the SOAP Gateway client. The IMS Connect HWSSOAP1 exit routine builds the required message structure containing the required internal headers for messages received from the client. HWSSOAP1 exit removes the internal headers for messages being transmitted to the client.

Related concepts

Ping support for IMS Connect (Communications and Connections)

WSDL-to-PL/I segmentation APIs exit routine (DFSPWSHK)

The IMS WSDL-to-PL/I segmentation API exit routine (DFSPWSHK) gets control during execution of DFSPWSIO APIs DFSXGETS, DFSXSETS, DFSQGETS, and DFSQSETS.

The DFSPWSIO APIs are used and referenced by the PL/I application templates that are generated by IBM Developer for System z® for the WSDL-to-PL/I top-down development scenario in IMS Enterprise Suite SOAP Gateway. You can use the DFSPWSHK exit routine to inspect, modify, or replace the buffer that contains the current SOAP header, body, or fault data structure before it is sent (DFSXSETS, DFSQSETS) or received (DFSXGETS, DFSQGETS). The programming pattern that is employed by this exit routine is that of an event handler where specific events are received from DFSPWSIO APIs (see the dfs_in_event_type parameter).

Because the DFSPWSIO APIs are invoked in both IMS Connect (DFSXSETS, DFSXGETS) and IMS message processing dependent regions (DFSQGETS, DFSQSETS), the DFSPWSHK exit routine must reside in a load library that is in STEPLIB of both dependent region types.

About this routine

The following table shows the attributes of the IMS WSDL-to-PL/I segmentation API exit routine.

Table 302. IMS WSDL-to-PL/I segmentation API exit routine attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL.
Naming convention	The user exit routine must be named DFSPWSHK.
Binding	This exit routine must be linked as serially reusable (REUS=SERIAL).
IMS callable services	This exit routine is not eligible to use IMS callable services.
Sample routine location	IMS.SDFSSRC distribution library.

This exit routine must be written as a FETCHABLE, EXTERNAL PL/I procedure and must not be linked with a module that contains a MAIN procedure.

Parameters

The following table describes the parameters to this exit routine.

Table 303. DFSPWSHK exit routine parameter list

Name	Declaration	Description
dfs_in_version	char(36) byaddr inonly	A read-only-string-by-reference that contains the version of the DFSPWSIO segmentation APIs that invoked this event handler. The version is represented as a 36-character GUID.
dfs_in_event_type	fixed bin(31) byvalue inonly	A read-only-integer-by-value that indicates the current event type for which this event handler is being invoked. Use the dfs_in_struct* and dfs_out_struct* parameters to inspect, modify, or replace the data structure buffer that is associated with the current event. The following event types are supported: • dfs_event_type=1 (DFSXSETS): This event type is reported immediately before a data structure buffer that was supplied on a call to the DFSXSETS API is segmented and copied into the IMS Connect output buffer as part of an IMS request message. • dfs_event_type=2 (DFSQGETS): This event type is reported immediately after a data structure buffer has been recovered from one or more segments of an IMS request message but before the data structure buffer is returned to the caller of the DFSQETS API. • dfs_event_type=3 (DFSQSETS): This event type is reported immediately before a data structure buffer that was supplied on a call to the DFSQSETS API is segmented and inserted into the IMS Message Queue as part of an IMS response message. • dfs_event_type=4 (DFSXGETS): This event type is reported immediately after a data structure buffer has been recovered from segments of an IMS response message but before the data structure buffer is returned to the caller of the DFSXGETS API.
dfs_in_namespace	wchar(1024) varying byaddr	A read-only-string-by-reference that contains the target namespace of the service that is associated with the current event.
dfs_in_service_name	wchar(512) varying byaddr inonly	A read-only-string-by-reference that contains the name of the service that is associated with the current event.
dfs_in_port_name	wchar(512) varying byaddr inonly	A read-only-string-by-reference that contains the name of the port that is associated with the current event.
dfs_in_operation_name	wchar(512) varying byaddr inonly	A read-only-string-by-reference that contains the name of the operation that is associated with the current event.
dfs_in_struct_type	fixed bin(31) byvalue inonly	A read-only-integer-by-value that indicates the type of the input structure that is associated with the current event. The following structure types are supported: • dfs_in_struct_type=1: ASOAP Header structure. • dfs_in_struct_type=2: A SOAP Body structure. • dfs_in_struct_type=3: A SOAP Fault structure.

Table 303. DFSPWSHK exit routine parameter list (continued)

Name	Declaration	Description
dfs_in_struct_name	wchar(100) varying byaddr inonly	A read-only-string-by-reference that contains the name of the structure that is associated with the current event.
dfs_in_struct_ptr	pointer byvalue inonly	A read-only-pointer-by-value to a buffer which contains the structure that is associated with the current event.
dfs_in_struct_size	fixed bin(31) byvalue inonly	A read-only-integer-by-value that provides the length in bytes of the structure that resides in the buffer pointed to by the dfs_in_struct_ptr parameter.
dfs_in_struct_state	fixed bin(31) byvalue inonly	A read-only-integer-by-value that indicates the state of the input structure buffer (see parameters dfs_in_struct*) that is associated with the current event. States other than the default are the result of previous calls to this event handler. The following states are supported: • dfs_in_struct_state=0 (default): The contents of the buffer have not been modified, nor has the buffer been replaced with a newly-allocated one. • dfs_in_struct_state=1: The contents of the buffer have been modified, but the buffer has not been replaced with a newly-allocated one. • dfs_in_struct_state=2: The contents of the buffer have not been modified, but the buffer has been replaced with a newly-allocated one. • dfs_in_struct_state=3: The contents of the buffer have been modified, and the buffer has been replaced with a newly-allocated one.
dfs_out_struct_ptr	pointer byaddr	A pointer-by-reference in which to write the address of a buffer that contains the modified or unmodified structure that is associated with the current event. The address can be that of the original buffer that is pointed to by the dfs_in_struct_ptr parameter. Or it can be that of a newly-allocated replacement buffer. The lifetime of the replacement buffer is managed by DFSPWSIO. Default value := dfs_in_struct_ptr **Restriction:** If Language Environment callable services are used to allocate a new buffer, then this module must be compiled with PL/I option CHECK(NOSTORAGE).

Table 303. DFSPWSHK exit routine parameter list (continued)

Name	Declaration	Description
dfs_out_struct_state	fixed bin(31) byaddr	An integer-by-reference in which to specify the state of the structure buffer that is pointed to by the dfs_out_struct_ptr parameter. This state represents an update to the state that is provided in the dfs_in_struct_state parameter. The following states are supported: • dfs_out_struct_state=0: The contents of the buffer have not been modified, nor has the buffer been replaced with a newly-allocated one. • dfs_out_struct_state=1: The contents of the buffer have been modified, but the buffer has not been replaced with a newly-allocated one. • dfs_out_struct_state=2: The contents of the buffer have not been modified, but the buffer has been replaced with a newly-allocated one. • dfs_out_struct_state=3: The contents of the buffer have been modified, and the buffer has been replaced with a newly-allocated one. Default value := dfs_in_struct_state

Related concepts

WSDL-to-PL/I segmentation APIs for adding business logic in generated PL/I templates (Application Programming)

Related reference

Include file DFSPWSH (Application Programming APIs)

IBM WebSphere DataPower message exit routine (HWSDPWR1)

The IBM WebSphere® DataPower® message exit routine provides IMS Connect support for DataPower SOA Appliances.

The HWSDPWR1 exit routine is shipped with IMS Connect and link-edited into the IMS.SDFSRESL data set. The source code for HWSDPWR1 is not shipped and cannot be modified or replaced.

This exit routine performs the same basic functions as the HWSSMPL1 exit routine, but with modifications to support the DataPower message format. The HWSDPWR1 exit adds the required internal message headers to incoming messages (received from a DataPower client) and removes them from outgoing messages (sent to a DataPower client).

IMS Connect OM Command exit routines (HWSCSLO0 and HWSCSLO1)

The IMS Connect OM Command exit routines (HWSCSLO0 and HWSCSLO1) are required by IMS Connect clients that submit IMS commands from a TCP/IP network to the Operations Manager (OM) component of the IMS Common Service Layer (CSL). IBM Management Console for IMS and Db2 for z/OS, for example, is an OM command client that requires the HWSCSLO1 exit routine.

HWSCSLO0 and HWSCSLO1 are delivered as object code only (OCO) with IMS Connect. The exit routines are formatted OCO to allow IMS Connect and the user message exit routines to synchronize without requiring simultaneous upgrades of other products when message exit functions change.

If your installation uses an IMS Connect client to communicate with OM, you must include the HWSCSLO0 or the HWSCSLO1 exit routine name in the EXIT= parameter of the TCPIP statement.

The HWSCSLO0 exit routine provides the following functions:

- Performs data translation of ASCII to EBCDIC for input messages.

- Performs data translation of EBCDIC to ASCII for output messages.
- Builds the IMS Connect message structure (BPE and OM headers required byIMS Connect) for input messages.
- Removes the IMS Connect internal OM headers for output messages.
- Defaults to COMMIT MODE=1.
- Defaults to SYNCH LEVEL=NONE.
- Analyzes the following message header options:
  - COMMIT MODE override of the default
  - SYNC LEVEL override of the default
  - If no client ID is passed to the exit, then the message exit generates the client ID

The HWSCSLO1 exit routine provides the following functions:

- Performs no translation for output messages.
- Builds the IMS Connect message structure (BPE and OM headers required by IMS Connect) for input messages.
- Removes the IMS Connect internal OM headers for output messages.
- Defaults to COMMIT MODE=1.
- Defaults to SYNCH LEVEL=NONE.
- Analyzes the following message header options:
  - COMMIT MODE override of the default
  - SYNC LEVEL override of the default
  - If no client ID is passed to the exit, then the message exit generates the client ID.

HWSCSLO0 and HWSCSLO1 exit routines are shipped with IMS Connect and bound into the IMS.SDFSRESL data set. You must use one or both of the exit routines to support IMS Connect clients that issue OM commands. The source code for HWSCLSO0 and HWSCSLO1 are not shipped and cannot be modified or replaced.

The COMMIT mode is set to 1, and the SYNC level is set to NONE. These values can be overridden by supplying either the COMMIT mode, the sync level, or both in the IRM_F2 and IRM_F3 fields in the user section of the IRM prefix of the message received from the IMS Connect client.

The HWSCSLO0 and HWSCSLO1 exit routines translate ASCII to EBCDIC and build the required message structure containing the required internal headers for messages received from the client. The HWSCSLO0 exit routine performs the translation from EBCDIC to ASCII and removes the internal headers for messages being transmitted to the client. The HWSCSLO1 exit routine does not perform any translation on the output data, but it does remove the internal headers for messages being transmitted to the client.

If you do not use an IMS Connect client that issues OM commands, you do not need to specify the HWSCSLO0 or HWSCSLO1 exit routine name in the TCPIP statement EXIT= parameter.

Related reference

Input message from client and passed to message exit (Communications and Connections)

Output message from message exit to client (Communications and Connections)

Format of user portion of IRM for HWSSMPL0, HWSSMPL1, and user-written message exit routines (Communications and Connections)

IMS Connect Port Message Edit exit routine

The IMS Connect Port Message Edit exit routine receives control between IMS Connect and the z/OS TCP/IP stack to modify the format of input and output messages when the message format required by the client application program is inconsistent with the format required by IMS Connect.

You might use the Port Message Edit exit routine if you have application programs that access IMS Connect from platforms that prevent the application programs from conforming to the standard message formats supported by IMS Connect.

The Port Message Edit exit routine is assigned to an individual, unique port. After a port is defined as using the Port Message Edit exit routine, all messages received and sent to the client on that port are passed to the exit routine.

On input, the Port Message Edit exit routine receives control after IMS Connect has received the complete message from TCP/IP, but before IMS Connect starts processing the message.

On output, the Port Message Edit exit routine receives control after IMS Connect has formatted the output message but before the message is sent to TCP/IP.

Attention: Do not issue any z/OS calls in the user message exit that result in an `MVS WAIT`. If the exit routine triggers an `MVS WAIT`, all work on the TCP/IP port stops until the WAIT is posted.

You specify a Port Message Edit exit routine as a 1- to 8-character name on the EDIT parameter of the PORT keyword in the TCP/IP configuration statement in the HWSCFGxx PROCLIB member.

The Port Message Edit exit routine runs as a type-2 BPE exit and can be managed by the BPE commands, such as **`DISPLAY USEREXIT`** and **`REFRESH USEREXIT`**. As a type-2 BPE exit routine, the Port Message Edit exit routine is defined to BPE dynamically and you do not need to define it during BPE configuration.

Entry to and exit from this exit routine are recorded as trace events for the IMS Connect Event Recorder exit routine (HWSTECL0).

IMS provides the source code for a sample Port Message Edit exit routine, HWSPIOX0.

The Port Message Edit exit routine is not supported for the following clients:

- IMS Universal database driver clients that use DRDA ports that are defined on the IMS Connect ODACCESS configuration statement.
- User-written DRDA source servers that use DRDA ports that are defined on the IMS Connect ODACCESS configuration statement.
- SSL clients that use a SSL port defined on the IMS Connect TCPIP configuration statement.
- Local option clients

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. On entry to the IMS Connect Port Message Edit exit routine, R1 points to a Standard BPE user exit parameter list. The field UXPL_EXITPLP in this list contains the address of the IMS Connect Port Message Edit exit routine parameter list (mapped by the HWSEXPIO macro). The registers contain the following contents:

Register	Contents
1	Address of BPE User Exit Parameter list (mapped by macro BPEUXPL)
13	Pointer to first of two pre-chained save areas
14	Return address
15	Entry point address

Parameter list

The following table describes the IMS Connect Port Message Edit exit routine parameters:

Table 304. IMS Connect Port Message Edit exit routine parameter list

Field	Offset	Length	Contents
PIOPRM_FUNCTION	X'00'	4	Call type: 'INIT' INITIALIZATION 'READ' INPUT FROM CLIENT 'XMIT' OUTPUT TO CLIENT 'TERM' TERMINATION
PIOPRM_FUNC_LVL	X'04'	1	Exit function level
PIOPRM_FUNC_BASE			X'01' Base function
PROPRM_FLAG1	X'05'	1	Flag byte
PIOPRM_FLG1UPD			X'80' Message was updated
PIOPRM_FLG1IPV6			X'40' IPV6 enabled, map with CLNT_IPV6
PIOPRM_RESV1	X'06'	2	Reserved
PIOPRM_PORT	X'08'	8	IMS Connect port number
PIOPRM_XIB	X'10'	4	Address of exit interface block
PIOPRM_RETCODE	X'14'	4	Return code
PIOPRM_RSNCODE	X18'	4	Reason code
PIOPRM_END_COMM	X'1C'	0	

The following table describes the IMS Connect Port Message Edit exit routine initialization parameters:

Table 305. IMS Connect Port Message Edit exit routine initialization parameter list

Field	Offset	Length	Contents
PIOPRM_BUFSIZE	X'1C	4	Buffer size required for updated message

The following table describes the IMS Connect Port Message Edit exit routine READ and XMIT parameters:

Table 306. IMS Connect Port Message Edit exit routine READ and XMIT parameter list

Field	Offset	Length	Contents
PIOPRM_ORG_BUF	X'1C'	4	Address of original message buffer
PIOPRM_ORG_SIZE	X'20'	4	Size of original message
PIOPRM_NEW_BUF	X'24'	4	Address of updated message buffer
PIOPRM_NEW_SIZE	X'28'	4	Size of updated message
PIOPRM_CLIENTID	X'2C'	0	Client ID structure
PIOPRM_CLNT_IPV4	X'2C'	8	IPV4 mapping
PIOPRM_CLNT_4FMLY	X2E''	2	Client family type
PIOPRM_CLNT_4PORT	X30''	2	Client port
PIOPRM_CLNT_4IPA	X'2C'	4	Client IP address
PIOPRM_CLNT_IPV6	X'2C'	28	IPV6 mapping

Table 306. IMS Connect Port Message Edit exit routine READ and XMIT parameter list (continued)

Field	Offset	Length	Contents
PIOPRM_CLNT_6LEN	X'2C'	1	Client socket length
PIOPRM_CLNT_6FMLY	X'2D'	1	Client family type
PIOPRM_CLNT_6PORT	X'2E'	2	Client port
PIOPRM_CLNT_6FLOW	X'30'	4	Client flow information
PIOPRM_CLNT_6IPA	X'34'	16	Client IP address
PIOPRM_CLNT_6SCOP	X'44'	4	Client scope ID

Contents of registers on exit

The exit must restore all registers on returning to the caller.

IMS Connect communications with user message exits

When the IMS Connect starts, it loads user message exits one at a time and calls each user message exit INIT subroutine.

Example: USREXIT1, USREXIT2, and USREXIT3 are defined in the HWSCFG parameter of the IMS Connect startup JCL as follows:

```
TCPIP=(HOSTNAME=...,EXIT=(USREXIT1,USREXIT2,USREXIT3),...)
```

IMS Connect loads USREXIT1 first and calls the USREXIT1 INIT subroutine. After successfully loading USREXIT1, IMS Connect loads USREXIT2 and calls the USREXIT2 INIT subroutine, and then repeats this process for USREXIT3. Any unsuccessful loading or INIT failure prevents IMS Connect from connecting with TCP/IP.

Important: If you define a user exit name in the IMS Connect configuration member, but that user exit cannot be loaded during IMS Connect startup, the job abends with Abend 806, RC=4.

In order to provide full user exit support in the IMS Connect environment, every user exit routine must include the subroutines INIT, READ, XMIT, TERM, and EXER. IMS Connect only supports assembler language exits.

When a user exit takes control, it saves the contents of the registers and restores them when returning to the caller. IMS Connect provides a 1 KB buffer in the parameter list to be used for this purpose.

Register contents on subroutine entry

The following table provides a brief description of the contents of the register on a subroutine entry.

Table 307. Register contents on subroutine entry

Register	Contents
1	Pointer to a parameter list that is defined in the HWSEXPRM macro.
14	Return address of IMS Connect.
15	Entry point address to the user exit routine. The entry point name and load module name for a user exit routine must be the same as the name used for the user exit routine in HWSCFG.

Register contents on subroutine exit

The following table provides a brief description of the contents of the register on a subroutine exit.

Table 308. Register contents on subroutine exit

Register	Contents
1	Pointer to a parameter list that is defined in the HWSEXPRM macro.

INIT subroutine

After a user exit has been successfully loaded, the INIT subroutine for that user exit is called and a parameter list is passed to that user exit.

Contents of parameter list pointed to by register 1 at INIT subroutine entry

The following table lists the contents of the parameter list that is passed to the user exit at entry.

Table 309. Contents of parameter list pointed to by register 1 at INIT subroutine entry

Field	Length	Meaning
EXPRM_FUNCTION	4 bytes	Character string of value INIT. Specifies that the function to be performed is: `Initialize user exit`.
EXPRM_TOKEN	4 bytes	Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and for local variables.
EXPRM_XIB	4 bytes	Address of XIB (exit interface block).

The user exit finishes all its initialization processes here. It returns two MSGID identifiers for the messages that it is to handle, as well as the increase to the output buffer size for its READ, XMIT, and EXER subroutines. The user exit returns the *increase* in buffer size, but not the actual buffer size. The only reason to return anything other than 0 is to allow the exit to add data to the data portion of the message. The storage required for the BPE headers and OTMA headers is computed by IMS Connect. Typically, one of the MSGIDs is used by ASCII clients and the other by EBCDIC clients. IMS Connect computes the actual size of the output buffer, and it allocates the buffer size before it passes control to the user exit for READ, XMIT and EXER. The two identifiers can take any value, in EBCDIC or ASCII, other than the reserved MSGIDs (see "Important," which follows), provided that the values are both unique among user exits called by a given IMS Connect. Blanks and binary 0 are significant. The IMS Connect saves these identifiers to identify the owner of the incoming request messages. Any conflict in the identifiers must be resolved before a TCP/IP connection can be made.

Important: The following MSGIDs are reserved.

HWSJAV
Supports IMS TM resource adapter clients

HWSCSL
Supports the IMSplex connection that uses the HWSCSLO0 user message exit

HWSCS1
Supports the IMSplex connection that uses the HWSCSLO1 user message exit

HWSOA1
Supports SOAP Gateway clients that use the HWSSOAP1 user exit

HWSDP1
Supports IBM® WebSphere® DataPower® SOA Appliance clients that use the HWSDPWR1 user exit

SAMPLE
Supports non-IMS TM resource adapter clients that use the HWSSMPL0 user exit

SAMPL1
Supports non-IMS TM resource adapter clients that use the HWSSMPL1 user exit

If duplicate MSGID identifiers exist, one of the user exits that uses the conflicting identifier must either be dropped or be rewritten with a unique identifier. A system administrator should coordinate the assignment of MSGIDs.

If the INIT subroutine fails to complete the initialization function successfully, the IMS Connect does not connect with TCP/IP. A system programmer can start the connection after the problem has been fixed by issuing the OPENPORT command. When all user exits have been loaded and initialized, the IMS Connect is ready to receive messages from TCP/IP application programs. The IMS Connect uses the TCP/IP Socket API to receive stream data across the net. The completion of a message is determined by its MSGLength value returned by TCP/IP to IMS Connect. The IMS Connect receives data up to the value specified in MSGLength and uses MSGID to determine which user exit receives control for processing the request message.

Contents of parameter list pointed to by register 1 at INIT subroutine exit

The following table lists the contents of the parameter list that is pointed to by Register 1 and then passed to the user exit during exit.

Table 310. Contents of parameter list pointed to by register 1 at INIT subroutine exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPINI_RETCODE	4 bytes	Binary. Specifies the return code, which can be one of the following: • 0=INIT function was successful. • 4=INIT function was not successful.
EXPINI_RSNCODE	4 bytes	Binary. Specifies the reason code.
EXPINI_STRING1	8 bytes	Character string. Specifies the first MSGID that clients can use to identify this user exit. This MSGID could be used for EBCDIC clients.
EXPINI_STRING2	8 bytes	Character string. Specifies the second MSGID that clients can use to identify this user exit. This MSGID could be used for ASCII clients.

Table 310. Contents of parameter list pointed to by register 1 at INIT subroutine exit (continued)

Field	Length	Meaning
EXPINI_BUFINC	4 bytes	Binary. Specifies the increase size to the output buffer needed to allow the exit to denote that data will be moved from the exit input buffer to the output buffer to add data to the message if required. Field EXPINI_BUFINC is an increased size for input and output messages above what is needed for the BPE and OTMA headers. If, for example, you want to have the exit add data to the message either on input or output, then there will be increase in buffer size.

READ subroutine

After a complete request message that originated at a TCP/IP client has been received, control is passed to the READ subroutine in the user exit whose `MSGID` matches the `MSGID` of that request message and a parameter list is passed to that user exit.

Subsections:

- "Contents of parameter list pointed to by register 1 at READ subroutine entry" on page 669
- "Contents of parameter list pointed to by register 1 at READ subroutine exit" on page 671

Contents of parameter list pointed to by register 1 at READ subroutine entry

The following table lists the contents of the parameter lists which are pointed to by Register 1 during the READ subroutine entry.

Table 311. Contents of parameter list pointed to by register 1 at READ subroutine entry

Field	Length	Meaning
EXPRM_FUNCTION	4 bytes	Character string of value READ. Specifies that the function to be performed is: `Read client data and convert it to OTMA format.`
EXPRM_TOKEN	4 bytes	Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables.
EXPRM_XIB	4 bytes	Address of XIB (exit interface block).
EXPREA_INBUF	4 bytes	Address of the input buffer.
EXPREA_IBUFSIZE	4 bytes	Binary. Specifies the size of the input buffer.
EXPREA_OUTBUF	4 bytes	Address of the output buffer.
EXPREA_OBUFSIZE	4 bytes	Binary. Specifies the size of the output buffer.

Table 311. Contents of parameter list pointed to by register 1 at READ subroutine entry (continued)

Field	Length	Meaning
EXPREA_FLAG1	1 byte	Data string flag: • X'80' - Input data contains a MSGID matching `EXPINI_STRING1.` • X'40' - Input data contains a MSGID matching `EXPINI_STRING2.`
EXPREA_FLAG2	1 byte	Data flag: • X'01' - Data moved by exit from INBUF to OUTBUF. • X'02' - If this EXPREA_IPV6 bit is turned on, IPV6 is enabled. Map EXPREA_SOCKET6 to AF-INET6 socket address structure. • X'04' - EXPREA_FL2SOCD indicates that Socket Descriptor is present in EXPREA_SOCDESC. • X'40' - First message indicator.
Reserved	2 bytes	Reserved space.
EXPREA_RACFID	8 bytes	Character string. Specifies the default user ID for RACF.
The following 28 bytes have two definitions: one definition is for a 4 byte IPV4 address (EXPREA_NAMEID) and another definition is for a 16 byte IPV6 address (EXPREA_SOCKET6).		
For a 4 byte IPV4 address:		
EXPREA_NAMEID	0 bytes	Pointer referenced to the next 16 bytes.
EXPREA_FAMILY	2 bytes	Binary. Specifies the client family type.
EXPREA_PORT	2 bytes	Binary. Specifies the client port number.
EXPREA_ADDRESS	4 bytes	Client's IP address.
EXPREA_RESERVE	8 bytes	Reserved space.
	12 bytes	Reserved.
For a 16 byte IPV6 address:		
EXPREA_SOCKET6	0 bytes	Map to the AF_INET6 socket address structure (if the EXPREA_IPV6 bit of EXPREA_FLAG2 is turned on).
EXPREA_6LEN	1 byte	Address of socket length.
EXPREA_6FAMILY	1 byte	Address of family.
EXPREA_6PORT	2 bytes	Port number used by the application.
EXPREA_6FLOW	4 bytes	Flow information.
EXPREA_6ADDR	16 bytes	INET address (NETID).
EXPREA_6SCOPE	4 bytes	Scope ID.

EXPREA_IBUFSIZE and EXPREA_OBUFSIZE are the sizes of the input buffer and output buffer, respectively. These sizes are not related to the actual length of the input data and output data. The input buffer contains an exact copy of the data that was received from the client. The user exit might need to perform an ASCII-to-EBCDIC conversion on the data so that the data can be properly interpreted by the IMS application. The user exit can use EXPREA_FLAG1 to determine where the data originated and whether additional processing is required by the exit.

IMS Connect also supplies the default RACF user ID and the client's TCP/IP connection information to the user exit. At this point, the user exit might edit or filter its client's input data, then translate that data to OTMA message segments and place them in the output buffer. The user exit also must specify the length of the output data in EXPREA_DATALEN.

Contents of parameter list pointed to by register 1 at READ subroutine exit

The following table lists the contents of the parameter list that are pointed to by Register 1 during the subroutine exit.

Table 312. Contents of parameter list pointed to by register 1 at READ subroutine exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPREA_RETCODE	4 bytes	Binary. Specifies the return code, which can be one of the following values: • 0=READ function was successful. Process the data. • 4=READ function was not successful. Send the data in EXPREA_OUTBUF back to client and disconnect the socket. • 8=READ function was not successful. Just clean up. • 20=READ function was successful. Keep socket connected.
EXPREA_RSNCODE	4 bytes	Binary. Specifies the reason code.
EXPREA_DATALEN	4 bytes	Binary. Specifies the size of data in the EXPREA_OUTBUF to be returned to IMS Connect. This field is only meaningful when EXPREA_RETCODE = 0 or 4.
EXPREA_UFLAG1	1 byte	User flag: • X'80' - Client requests IMS MOD name be returned
Reserved	3 bytes	Reserved space.
EXPREA_CLID	8 bytes	Character string. It specifies the client ID name passed by the client or generated by the exit for non-IMS TM Resource Adapter clients only.
EXPREA_SVT	4 bytes	Address of SVT.
EXPREA_LSTNPORT	2 bytes	Binary. Specifies the listening port number.

The output buffer contains data when the return code is 0 or 4. When the return code is 4, the data in the output buffer is sent back to the user exit's client, and then the connection is closed and cleaned up. When the return code is 0, IMS Connect prepares to present the data to a data store. EXPREA_UFLAG1 is also saved by IMS Connect. This flag is set by the user exit during READ subroutine processing and is used for recording user selected characteristics of the request message. This flag is passed back to the user exit in the input parameter list pointed to by Register 1 on the next subroutine call, which is either an XMIT or an EXER subroutine call. You define the value of EXPREA_UFLAG1 in the user exit code. IMS Connect uses this value to provide a communication vehicle between the READ and XMIT or EXER subroutines on a per request/response message basis. The XMIT and EXER subroutines can thus format the message in a better manner.

If IMS Connect detects an error in the output data that would prevent it from properly presenting the data to the data store (for example, the output data is not formatted properly to conform to the IMS OTMA protocol), the EXER subroutine is called where the error can be dealt with appropriately. IMS Connect then waits until it receives the response message from IMS OTMA. After receiving a response, it calls the XMIT subroutine of the appropriate user exit (based on the MSGID in the response) and passes it an exact copy of the response data that it received from IMS OTMA.

XMIT subroutine

After a complete response message has been received from the data store, control is passed to the XMIT subroutine in the user exit whose MSGID matches the MSGID of the response message (which in turn matches the MSGID of the original request message) and a parameter list is passed to that user exit.

Subsections:

- "Contents of parameter list pointed to by register 1 at XMIT subroutine entry" on page 672
- "Contents of parameter list pointed to by register 1 at XMIT subroutine exit" on page 673

Contents of parameter list pointed to by register 1 at XMIT subroutine entry

The following table lists the contents of the parameter list that are pointed to by Register 1 during the XMIT subroutine entry and passed to the user exit.

Table 313. Contents of parameter list pointed to by register 1 at XMIT subroutine entry

Field	Length	Meaning
EXPRM_FUNCTION	4 bytes	Character string of value XMIT. Specifies that the function to be performed is: `Read OTMA data and convert it to client format.`
EXPRM_TOKEN	4 bytes	Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables.
EXPRM_XIB	4 bytes	Address of XIB (exit interface block).
EXPXMT_INBUF	4 bytes	Address of the input buffer.
EXPXMT_IBUFSIZE	4 bytes	Binary. Specifies the size of the input buffer.
EXPXMT_OUTBUF	4 bytes	Address of the output buffer.
EXPXMT_OBUFSIZE	4 bytes	Binary. Specifies the size of the output buffer.

Table 313. Contents of parameter list pointed to by register 1 at XMIT subroutine entry (continued)

Field	Length	Meaning
EXPXMT_FLAG1	1 byte	Data string flag: • X'80' - Input data contains a MSGID matching EXPINI_STRING1. • X'40' - Input data contains a MSGID matching EXPINI_STRING2. • X'20' - EXPXMT_F1_SYNC indicates a synchronous callout message. • X'02' - EXPXMT_F1_CLID indicates EXPXMT_CLID contains the Client_id name.
EXPXMT_UFLAG1	1 byte	User flag. X'xx' - User-defined value. The value was set in READ subroutine.
Reserved	2 bytes	Reserved space.
EXPRM_CLID	8 bytes	Character. Client_id name assigned to sessions.

EXPXMT_IBUFSIZE and EXPXMT_OBUFSIZE are the sizes of the input buffer and output buffer, respectively. These sizes are not related to the actual length of the input data and output data. The input buffer contains an exact copy of the OTMA message segments that were received from the data store. The user exit might need to perform an EBCDIC-to-ASCII conversion on the data so that the data can be properly interpreted by the client application. The user exit translates OTMA message segments to its client's data format, places the data in the output buffer, and specifies the length of the output data in EXPXMT_DATALEN. The user exit might also edit or filter the output data at this point.

Contents of parameter list pointed to by register 1 at XMIT subroutine exit

The following table lists the contents of the parameter list that are pointed to by Register 1 during the XMIT subroutine exit.

Table 314. Contents of parameter list pointed to by register 1 at XMIT subroutine exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPXMT_RETCODE	4 bytes	Binary. Specifies the return code, which can be one of the following values: • 0=XMIT function was successful. Process the data. • 8=XMIT function was not successful. Just clean up.
EXPXMP_RSNCODE	4 bytes	Binary. Specifies the reason code.
EXPXMT_DATALEN	4 bytes	Binary. Specifies the size of data in the EXPXMT_OUTBUF to be returned to IMS Connect. This field is only meaningful when EXPXMT_RETCODE = 0.

When the return code is 0, the data in the output buffer is sent back to the originator of the client request message. If the return code is not 0, the connection is dropped. If the user exit sets a non-zero return code value, the connection closes without sending a response back to the originator of the client request message.

TERM subroutine

When IMS Connect is shutting down, control is passed, in turn, to the TERM subroutine in each user exit that is currently active, and a parameter list is passed to that user exit.

Subsections:

- "Contents of parameter list pointed to by register 1 at TERM subroutine entry" on page 674
- "Contents of parameter list pointed to by register 1 at TERM subroutine exit" on page 674

Contents of parameter list pointed to by register 1 at TERM subroutine entry

The following table lists the contents of the parameter list that are pointed to by Register 1 during TERM Subroutine entry and passed to the user exit.

Table 315. Contents of parameter list pointed to by register 1 at TERM subroutine entry

Field	Length	Meaning
EXPRM_FUNCTION	4 bytes	Character string of value TERM. Specifies that the function to be performed is: `Clean up in preparation for IMS Connect shutdown.`
EXPRM_TOKEN	4 bytes	Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables.
EXPRM_XIB	4 bytes	Address of XIB (exit interface block).

The user exit finishes all its termination processes here.

IMS Connect shutdown proceeds independently of the return code value. The return code merely indicates the completeness of the user exit cleanup.

Contents of parameter list pointed to by register 1 at TERM subroutine exit

The following table lists the contents of the parameter list that are pointed to by Register 1 during the TERM subroutine exit.

Table 316. Contents of parameter list pointed to by register 1 at TERM subroutine exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPTRM_RETCODE	4 bytes	Binary. Specifies the return code, which can be one of the following values: • 0=TERM function was successful. • 4=TERM function was not successful.

Table 316. Contents of parameter list pointed to by register 1 at TERM subroutine exit (continued)

Field	Length	Meaning
EXPTRM_RSNCODE	4 bytes	Binary. Specifies the reason code. The reason codes are set by the exits (HWSSMPL0, HWSSMPL1, and HWSJAVA0).

EXER subroutine

When IMS Connect detects an error in the output buffer after execution of the previous READ subroutine completes, control is passed to the EXER subroutine in the same user exit where the READ subroutine executed and a parameter list is passed to that user exit.

Subsections:

- "Contents of parameter list pointed to by register 1 at EXER subroutine entry" on page 675
- "Contents of parameter list pointed to by register 1 at EXER subroutine exit" on page 676

Contents of parameter list pointed to by register 1 at EXER subroutine entry

The following table lists the contents of the parameter list that are pointed to by Register 1 during EXER subroutine entry and passed to the user exit.

Table 317. Contents of parameter list pointed to by register 1 at EXER subroutine entry

Field	Length	Meaning
EXPRM_FUNCTION	4 bytes	Character string of value EXER. Specifies that the function to be performed is: `Process error found in output buffer after previous READ subroutine processing completed.`
EXPRM_TOKEN	4 bytes	Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables.
EXPRM_XIB	4 bytes	Address of XIB (exit interface block).
EXPXER_OUTBUF	4 bytes	Address of the output buffer.
EXPXER_OBUFSIZE	4 bytes	Binary. Specifies the size of the output buffer.
EXPXER_FLAG1	1 byte	Data string flag, which can be one of the following values: • X'80' - Input data contains a MSGID matching `EXPINI_STRING1.` • X'40' - Input data contains a MSGID matching `EXPINI_STRING2.`
EXPXER_UFLAG1	1 byte	User flag. X'xx' - User-defined value. The value was set in READ subroutine.
Reserved	2 bytes	Reserved space.
EXPXER_CODE	4 bytes	Binary. Specifies the failure code. • 4=Error in the output buffer from the previous READ function.

Table 317. Contents of parameter list pointed to by register 1 at EXER subroutine entry (continued)

Field	Length	Meaning
EXPXER_REASON	4 bytes	Binary. Specifies the failure reason, which can be one of the following: • 20=Segment length error • 24=Missing first in chain flag • 28=Missing last in chain flag • 32=Sequence number error

The user exit could have experienced difficulties in forming OTMA message segment format and should notify its client of this situation (for example, through an error message). The user exit can use EXPXER_FLAG1 to determine where the request message from the client originated and whether to compose an ASCII or EBCDIC data stream for sending back to the originating client.

Contents of parameter list pointed to by register 1 at EXER subroutine exit

The following table lists the contents of the parameter list that are pointed to by Register 1 during the EXER subroutine exit.

Table 318. Contents of parameter list pointed to by register 1 at EXER subroutine exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPXER_RETCODE	4 bytes	Binary. Specifies the return code, which can be one of the following values: • 4=Send the data in EXPXER_OUTBUF back to client. • 8=Just clean up.
EXPXER_RSNCODE	4 bytes	Binary. Specifies the reason code.
EXPXER_DATALEN	4 bytes	Binary. Specifies the size of data in the EXPXER_OUTBUF to be returned to clients. This field is only meaningful when EXPER_RETCODE=4.

When the return code is 4, IMS Connect sends the data in the output buffer back to the client. If the user exit sets the return code value to 8, the connection closes without a response.

Macros that support IMS Connect user message exits

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.

Chapter 16. IMS Connect function-specific exit routines

IMS provides several exit routines with IMS Connect for additional flexibility.

IMS Connect User Initialization exit routine (HWSUINIT)

The IMS Connect User Initialization exit routine (HWSUINIT) can perform customized tasks during IMS Connect startup, IMS Connect shutdown, or both.

For example, you can modify the HWSUINIT exit routine to display a specific message when IMS Connect starts up or shuts down.

The HWSUINIT routine contains two user control blocks that enable further customization: XIB and XIBDS. The XIB control block can be used to store any data that you want. The XIBDS control block keeps track of the status of the IMS Connect data stores. All of the IMS Connect user message exits can access both the XIB and XIBDS user control blocks.

For example, you can modify HWSUINIT to load a specific table when IMS Connect starts up, then store the table address into the XIB control block area. After the IMS Connect user message exits get control, they access that table and perform their customized processing. When IMS Connect shuts down, you can modify HWSUINIT to unload the updated table.

The HWSUINIT user initialization exit routine that comes with IMS Connect does not do any processing. HWSUINIT is provided as a load module for ease of use. Source code is also provided for modification, but you must assemble and link edit the source to use a modified version. Modify HWSUINIT only if you want to use it.

How IMS Connect communicates with HWSUINIT

HWSUINIT contains two subroutines: INIT and TERM. When IMS Connect starts, HWSUINIT loads and gives control to the INIT subroutine. When IMS Connect shuts down, HWSUINIT gives control to the TERM subroutine.

HWSUINIT contains two of its own user control blocks: XIB and XIBDS. The HWSXIB and HWSXIBDS DSECTs map the XIB and XIBDS user control blocks. The message exit routines in the INIT, READ, XMIT, TERM, and EXER subroutines can also use the XIB and XIBDS user control blocks. The XIB user control block contains a fixed length header section and a variable length user area.

Restriction: You cannot modify the fixed header section. You can only modify the user area.

You specify the size of the XIB control block user area, in full words, with the *xibarea* parameter (in the HWS statement of the IMS Connect configuration file). The default value is 20; the maximum value is 500. If you do not specify a value for the *xibarea* parameter, or you specify a value outside of the 20 to 500 range, IMS Connect uses the default value of 20.

The XIBDS user control block represents an entry in a list of data stores that are defined in the configuration file. The second word in the fixed header area of the XIB user control block points to the data store list. The XIBDS user control block is 16 bytes long. Each data store list entry contains the data store name, the data store status (active or inactive), a flag byte, and a 4 byte field that you can use to store any kind of data. The last entry is indicated by a value of X'80' (hexadecimal) in the flag byte. The number of entries in the list is equal to the number of data stores defined in the IMS Connect configuration file.

Because the XIBDS user control block keeps track of *all* IMS Connect data store statuses, you can enable any user message exit to take action based on the status of one or more of the IMS Connect data stores. For example, before a user message exit passes a client message to an IMS Connect data store for processing, you could have the user message exit query the XIBDS control block area for the target data

store's status. If the target data store is not active, you could enable the user message exit to switch to an active data store by modifying the data store name in the message header.

When the HWSUINIT routine takes control, it saves the contents of the registers and restores them when returning to the caller. IMS Connect provides a 1 KB buffer in the parameter list to be used for this purpose.

Register contents on HWSUINIT entry

The following table lists the contents of each register on the HWSUINIT entry.

Table 319. Register contents on HWSUINIT entry

Register	Contents
1	Pointer to a parameter list: • +0 — XIB address • +4 — Function to perform (INIT or TERM) • +8 — 1 KB buffer for exit to use
14	Return address of IMS Connect.
15	Entry point address to HWSUINIT.

Register contents on HWSUINIT exit

The following table lists the contents of each register on the HWSUINIT exit.

Table 320. Register contents on HWSUINIT exit

Register	Contents
0–14	Restored.
15	0 — completed successfully. 1 to 7 — warning, but IMS Connect initialization continues. 8 or higher — force IMS Connect termination.

Related reference

“Macros that support IMS Connect user message exits” on page 676
IMS provides macros that support the IMS Connect exit routines.

IMS Connect User Initialization exit routine (HWSUINIT) sample JCL

You can use JCL to modify HWSUNIT that will perform processing, such as customized initialization tasks during IMS Connect startup, customized termination tasks during IMS Connect shutdown, or both.

For an example of an HWSUINIT user initialization exit that performs processing, see the following JCL.

```
//HWSINIT JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//UINIT1 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE),SYSPARM(HWSUINIT)'
//SYSLIB DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSUINIT)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMS.SDFSSMPL(xxxxxx),DISP=SHR
```

```
//UINIT2 EXEC PGM=IEWL,
//        PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSUINIT)
 ENTRY   HWSUINIT
 NAME HWSUINIT(R)
//
```

IMS Connect DB Routing user exit routine (HWSROUT0)

Use the IMS Connect DB Routing user exit routine (HWSROUT0) to control the routing of messages from IMS DB clients, such as the IMS Universal drivers, to specific instances of IMS or the Open Database Manager (ODBM).

The HWSROUT0 user exit routine can override the IMS alias name specified by the IMS Connect client. If the HWSROUT0 user exit routine overrides the IMS alias name, IMS Connect uses the IMS alias name specified by the exit routine.

The HWSROUT0 user exit routine can also select a specific instance of the CSL Open Database Manager (ODBM) to which to route an incoming message.

After the HWSROUT0 user exit routine returns the message and control to IMS Connect, IMS Connect routes the message based on the alias name or the ODBM instance name specified in the message.

If the HWSROUT0 user exit routine selects an ODBM, IMS Connect uses that ODBM and will not perform the round robin routing method.

If the HWSROUT0 user exit routine does not select an ODBM, the IMS alias name determines how IMS Connect selects the ODBM instance to deliver the message to. If an alias name is specified, IMS Connect routes the message to the ODBM instance that supports the alias name. If multiple ODBM instances support the alias name, IMS Connect uses a round-robin algorithm to distribute incoming messages among the ODBM instances. If the IMS alias name is blank, IMS Connect uses a round-robin algorithm to distribute the messages among all active ODBM instances.

IMS Connect validates the alias name and the ODBM instance after the exit routine returns control to IMS Connect.

The HWSROUT0 user exit routine runs as a BPE type-1 exit routine and must conform to the BPE type-1 interface. The HWSROUT0 user exit routine can be managed with the BPE DISPLAY USEREXIT and REFRESH USEREXIT commands. The HWSROUT0 user exit routine is also passed the standard BPE user exit parameter list that is mapped by the BPEUXPL macro. The exit-type-specific parameter list (UXPL_EXITPLP) points to the HWSROUT0 exit parameter list (HWSROUPM).

Note: Do not issue any MVS calls in the user message exit that result in an `MVS WAIT` because the `MVS WAIT` will halt all work on the port. If you modify the exit routine 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 exit routine cannot be modified to free any storage passed to the exit routine, and IMS Connect will not free any storage obtained by the exit routine when the exit routine 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 routine without causing failures.

To use the HWSROUT0 user exit routine, perform the following basic steps:

1. Create a new or modify an existing BPE exit list PROCLIB member with any name, for example HWSEXIT0.
2. In the BPE exit list PROCLIB member, specify the following EXITDEF statement:

   ```
   EXITDEF(TYPE=ODBMROUT,EXITS=(exitname),ABLIM=limit,COMP=HWS)
   ```

   where *exitname* is the name of the HWSROUT0 user exit routine and *limit* is the number of times the exit can abend before it is disabled.

3. Set the BPE exit list PROCLIB member in the BPE configuration parameter PROCLIB member by adding an EXITMBR statement. For example, if the BPE exit list PROCLIB member is HWSEXIT0, then add the following statement to the BPE configuration member:

```
EXITMBR=(HWSEXIT0,HWS)     /* IMS CONNECT EXITS */
```

Contents of register on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the "Standard BPE user exit parameter list" on page 485, mapped by macro BPEUXPL
13	Save area address
14	Return address
15	Entry point address

On entry to the IMS Connect DB routing user exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the IMS Connect DB routing user exit parameter list, which is mapped by macro HWSROUPM. Field UXPL_COMPTYPEP in this list points to the character string "HWS" indicating an IMS Connect address space.

Parameter list

Table 321. HWSROUT0 user exit routine parameter list

Field name	Offset	Length	Description
ROUPM_PVer	X'00'	X'04'	Version number of the parameter list: **1** Base version for IMS Version 11. **2** Current version for IMS Version 11. Introduced with APAR PM22144. **17** Base and current version for IMS Version 12 and later.
ROUPM_Function	X'04'	X'04'	Function type: **INIT** Initialization **ROUT** Routing **TERM** Termination
ROUPM_Aclstruct	X'08'	X'04'	Address of the client ID structure

Table 321. HWSROUT0 user exit routine parameter list (continued)

Field name	Offset	Length	Description
ROUPM_Flag1	X'0C'	X'01'	Flag byte: **X'80'** EBCDIC encoding **X'40'** IPv6 client IP address **X'20'** Client request for ODBM z/OS Resource Recovery Services
	X'0D'	X'07'	Reserved
ROUPM_AUsrdataL	X'14'	X'04'	Address of the user data length
ROUPM_AUsrdata	X'18'	X'04'	Address of the DRDA user data
ROUPM_AInAlias	X1C'	X'04'	Address of the 4-character IMS alias name
ROUPM_AInPsbnm	X'20'	X'04'	Address of the 8-character PSB name
ROUPM_Xib	X'24'	X'04'	XIB address
ROUPM_AOutlias	X'28'	X'04'	Address of the 4-character IMS alias set by the exit (output)
ROUPM_AOutOdbm	X'2C'	X'04'	Address of the 8-character ODBM name set by the exit (output)
ROUPM_ARetcode	X'30'	X'04'	Address of the fullword return code set by the exit (output)
ROUPM_ARsncode	X'34'	X'04'	Address of the fullword reason code set by the exit (output)

Contents of register on exit

The exit must restore all registers on returning to the caller.

Related reference

“Macros that support IMS Connect user message exits” on page 676
IMS provides macros that support the IMS Connect exit routines.

BPE exit list members of the IMS PROCLIB data set (System Definition)

IMS Connect DB security user exit routine (HWSAUTH0)

You can use the IMS Connect DB security user exit routine to authenticate the input user ID and password specified by IMS Connect clients that access IMS DB, such as any of the IMS Universal drivers or a client application program that connects to IMS Connect through a user-written DRDA source server.

IMS Connect always calls the HWSAUTH0 user exit routine before invoking any installed security facility, such as RACF, if one is enabled.

The HWSAUTH0 user exit routine can override the input user ID with a different user ID. The HWSAUTH0 user exit routine can provide a RACF group ID to be authenticated further by IMS Connect.

The HWSAUTH0 user exit routine is refreshable.

The HWSAUTH0 user exit routine runs as a BPE type-1 exit routine and must conform to the BPE type-1 interface. The HWSAUTH0 user exit routine can be managed with the BPE DISPLAY USEREXIT and REFRESH USEREXIT commands. The HWSAUTH0 user exit routine is passed the standard parameter list for BPE user exit routines, which is mapped by the BPEUXPL macro. The exit-type-specific parameter list (UXPL_EXITPLP) points to the HWSAUTH0 exit parameter list (HWSAUTPM).

Note: Do not issue any MVS calls in the user message exit that result in an MVS WAIT because the MVS WAIT will halt all work on the port. If you modify the exit routine 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 exit routine cannot be modified to free any storage passed to the exit routine, and IMS Connect will not free any storage obtained by the exit routine when the exit routine 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 routine without causing failures.

To use the HWSAUTH0 user exit routine, perform the following basic steps:

1. Create a new or modify an existing BPE exit list PROCLIB member with any name, for example HWSEXIT0.
2. In the BPE exit list PROCLIB member, define HWSAUTH0 as an exit in the following EXITDEF statement:

   ```
   EXITDEF(TYPE=ODBMAUTH,EXITS=(HWSAUTH0),ABLIM=8,COMP=HWS)
   ```

   All of the parameters must be coded as shown except for ABLIM, which sets the number of times the exit can abend before it is disabled.
3. Set the BPE exit list PROCLIB member in the BPE configuration parameter PROCLIB member by adding an EXITMBR statement. For example, if the BPE exit list PROCLIB member is HWSEXIT0, then add the following statement to the BPE configuration member:

   ```
   EXITMBR=(HWSEXIT0,HWS)     /* IMS CONNECT EXITS */
   ```

HWSAUTH0 security exit is shipped with IMS Connect and link-edited into the IMS.SDFSRESL data set.

Contents of registers on entry

On entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the “Standard BPE user exit parameter list” on page 485. The UXPL_EXITPLP field in this parameter list contains the adress of the IMS Connect DB security user exit parameter list, which is maped by macro HWSAUTPM.
13	Save area address
14	Return address
15	Entry point address

Parameter list

Table 322. HWSAUTH0 user exit routine parameter list

Field name	Offset	Length	Description
AUTPM_PVer	X'00'	X'04'	Version number of the parameter list: **1** Base version for IMS Version 11. **2** Current version for IMS Version 11. In IMS Version 11, added the **AUTPM_Aclstruct** field. Introduced with APAR PM22144. **17** Base and current version for IMS Version 12 and later. **33** Reserved for IMS Version 13. **49** Added the **AUTPM_AAppl** field in IMS 14. **50** Current version for IMS 14. Added the **AUTPM_APsswordL** field in IMS 14. **65** Added the **AUTPM_AAppl** field in IMS 15. **66** Current version for IMS 15. Added the **AUTPM_APsswordL** field in IMS 15.
AUTPM_Aclstruct	X'04'	X'04'	Address of the client's ID structure
AUTPM_Flag1	X'08'	X'01'	Flag byte: **X'80'** EBCDIC encoding **X'40'** IPv6 client IP address **X'20'** **AUTPM_APsswordL** field can be used
	X'09'	X'07'	Reserved
AUTPM_AusrDataL	X'10'	X'04'	Address of user data length
AUTPM_AusrData	X'14'	X'04'	Address of DRDA user data
AUTPM_AIUserid	X'18'	X'04'	Address of user ID
AUTPM_APssword	X1C'	X'04'	Address of password or passphrase
AUTPM_ARetcode	X'20'	X'04'	Address of fullword return code set by the exit (output)
AUTPM_ARsncode	X'24'	X'04'	Address of the fullword reason code set by the exit (output) Reason code 0 - if return code is 0 Reason code ¬0 - if return code is 4
AUTPM_AOUserid	X'28'	X'04'	Address of the 8-character user ID set by the exit (output)
AUTPM_AOGrpid	X'2C'	X'04'	Address of the 8-character group ID set by the exit (output)

Table 322. HWSAUTH0 user exit routine parameter list (continued)

Field name	Offset	Length	Description
AUTPM_AAppl	X'30'	X'04'	Address of the 8-character application name.
AUTPM_APsswordL	X'34'	X'04'	Address of a fullword that contains the length (1 - 100) of the password or the password phrase. This password or password phrase is pointed to by the **AUTPM_APssword** field. The **AUTPM_APsswordL** is available only if the **AUTPM_Pwlen_Set** field is enabled.

Contents of registers on exit

The exit must restore all registers on returning to the caller.

Using the IMS Connect DB security user exit routine

The IMS Connect DB security user exit routine (HWSAUTH0) user exit routine runs as a BPE Type-1 exit and must conform to that interface. HWSAUTH0 can be managed with the **BPE DISPLAY USEREXIT** and **REFRESH USEREXIT** commands.

The exit is passed the Standard BPE User Exit Parameter List (mapped by the BPEUXPL macro). The exit-type-specific parameter list (UXPL_EXITPLP) points to the HWSAUTH0 exit parameter list (HWSAUTPM). If you want to use this exit you must perform the following steps:

1. Create or modify a BPE exit list PROCLIB member with any name, for example HWSEXIT0.
   In the BPE exit list PROCLIB member, define HWSAUTH0 in the following EXITDEF statement: `EXITDEF(TYPE=ODBMAUTH,EXITS=(HWSAUTH0),ABLIM=8,COMP=HWS)`. All parameters must be coded as shown except for ABLIM, which sets the number of times the exit can abend before it is disabled.
2. Set the BPE exit list PROCLIB member in the BPE configuration parameter PROCLIB member by adding an EXITMBR statement. For example, if the BPE exit list PROCLIB member is HWSEXIT0, then add the following statement to the BPE configuration member:

```
EXITMBR=(HWSEXIT0,HWS) /* IMS CONNECT EXITS */
```

IMS Connect sample OTMA User Data Formatting exit routine (HWSYDRU0)

An OTMA User Data Formatting exit routine (DFSYDRU0) is required to support asynchronous output that is generated by an IMS application that does an insert (ISRT) to an alternate PCB (program communication block).

IMS Connect provides a sample OTMA User Data Formatting exit routine named HWSYDRU0. You can either modify the HWSYDRU0 exit routine to work with your installation, or provide your own OTMA User Data Formatting exit routine. Regardless of which you use, the OTMA Destination Resolution exit routine runs in the IMS control region and not in the IMS Connect address space.

How IMS Connect communicates with the DRU exit

OTMA allows transaction pipe names (TPIPEs) to be the same as an IMS LTERM name. In IMS Connect, the LTERM name is analogous to the unique CLIENTID name. To clarify whether a destination is for IMS Connect (through OTMA), IMS provides OTMA exit routines that can specify where IMS should look to resolve the destination names. In this case, the IMS needs to look at the IMS Connect CLIENTIDs. The DRU exit cannot change the actual destination name. Determining the destination for an OTMA (IMS Connect client) message requires two phases.

1. The OTMA 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 OTMA (IMS Connect clients) or to IMS TM for processing. The user exit cannot determine the final destination.
2. The DRU exit routine (for example, the IMS Connect supplied exit HWSYDRU0) is called to determine the final destination for the output.

   Each OTMA client can specify a separate DRU exit routine. In other words, each OTMA client can specify a single DRU exit for each copy of IMS Connect that is connected to a given data store (IMS). This means that one IMS Connect can have the same or a different DRU exit for each of the data store definitions in the IMS Connect configuration file.

How to use the HWSYDRU0 exit

HWSYDRU0, the IMS Connect supplied OTMA DRU exit, provides only a sample of what the DRU exit can do. You can use this exit only under one of the following conditions:

- The IMS Connect CLIENTIDs are named CLIENT01 through CLIENT09 and they all belong to the same member name.
- The non-IMS Connect CLIENTIDs are as follows:
  - TPIPE001 through TPIPE099 all belong to member MEMBER0
  - TPIPE100 through TPIPE199 all belong to member MEMBER1
  - TPIPE200 through TPIPE299 all belong to member MEMBER2
  - TPIPE300 through TPIPE399 all belong to member MEMBER3
  - TPIPE400 through TPIPE499 all belong to member MEMBER4
  - TPIPE500 through TPIPE599 all belong to member MEMBER5
  - TPIPE600 through TPIPE699 all belong to member MEMBER6
  - TPIPE700 through TPIPE799 all belong to member MEMBER7
  - TPIPE800 through TPIPE899 all belong to member MEMBER8
  - TPIPE900 through TPIPE999 all belong to member MEMBER9

The HWSYDRU0 exit is only an example, and when you use it, the following sequence of events will occur:

1. The OTMA Destination Resolution user exit (OTMAYPRX) sets up addressability to the parameters that are passed to the HWSYDRU0 exit.
2. The output member name in the output parameter list is set to blanks.
3. HWSYDRU0 determines the action to take based on whether the name in the input destination parameter (that is, the destination where the message is to be sent) is an IMS LTERM or an IMS Connect destination. After HWSYDRU0 makes this determination, it takes a course of action, and sets the contents of register 15 on exit.
4. If an IMS application was initiated by a non-IMS Connect client, then HWSYDRU0 must build the OTMA user data.
5. If HWSYDRU0 places the character string, ICONNECT, into the OTMA user data header field, OMUSR_PORTID, (whether built by HWSYDRU0 or passed to HWSYDRU0) then IMS Connect will determine the correct PORTID to be used for the selected output client ID.

The following table describes the register settings and the action taken for the specific return code.

Table 323. Register settings and HWSYDRU actions

Register settings	HWSYDRU actions
Register 15 = X'00'	• The input destination name is an IMS Connect client name and the member name for the destination is the same as the member name for the origin. • No changes made to the output parameters.
Register 15 = X'04'	• LTERM exists in IMS (LEGACY), is not an IMS Connect client. • No changes made to the output parameters.
Register 15 = X'08'	• The input destination name is an IMS Connect client name, and the member name for the destination is a different name from the member name for the origin. • The output member name in the output parameters is set to the new member name.
Register 15 = X'0C'	The input destination name is not an LTERM for IMS, and IMS Connect does not know the client name.

IMS Connect sample OTMA User Data Formatting (HWSYDRU0) sample JCL

You can use JCL to modify the HWSYDRU0 exit to work with your installation.

To review how JCL can be modified, refer to the following HWSYDRU0 sample OTMA DRU exit.

```
//HWSYDRU JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//YDRU01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE),SYSPARM(HWSYDRU0)'
//SYSLIB DD DSN=IMS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSYDRU0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMS.SDFSSMPL(xxxxxx),DISP=SHR
//YDRU02 EXEC PGM=IEWL,
//        PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSYDRU0)
 ENTRY   HWSYDRU0
 NAME HWSYDRU0(R)
//
```

z/OS TCP/IP IMS Listener security exit (IMSLSECX)

If any IMS Connect user message exit 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 due to the many options available for security and the fact that most installations have their own specific security method.

The call to RACF or other security product is performed by IMS Connect if RACF parameters are provided in the OTMA header when the user message exit routine returns the message to IMS Connect.

By default, IMSLSECX is the name of the security exit routine called by the following IMS Connect user message exit routines:

- HWSSMPL0
- HWSSMPL1
- HWSSOAP1
- HWSCSLO0

You can also define the name of the security exit called by HWSJAVA0 in the HWSJAVA0 message exit routine.

If you use HWSSMPL0 or HWSSMPL1, you can change the name of the security exit that is called 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.

Parameter list for user security exit

Following is the list and order of parameters being passed to the security exit, IMSLSECX. The order of the parameters is fixed for the exits supplied by IMS Connect: HWSSMPL0 and HWSSMPL1. The parameters are mapped in the HWSIMSEA macro at IMSEA_SecParml.

- Address of fullword client's IP address
- Address of halfword client's port
- Address of 8-char string IMS transaction
- Address of halfword data type (data type setting: 0=ASCII, 1=EBCDIC)
- Address of fullword length of user data
- Address of user-supplied data
- Address of fullword set by security exit
- Address of fullword set by security exit
- Address of RACF user ID

  If blanks are returned (in the field pointed to) from the security exit, then the RACF fields in the OTMA security header are not set.

  The address points to a field containing blanks.
- Address of RACF group ID

  The address points to a field containing blanks.

Related reference

“IMS TM Resource Adapter user message exit routine (HWSJAVA0)” on page 657
Use the IMS TM Resource Adapter user message exit routine (HWSJAVA0) to edit messages and perform custom security checking in support of the IMS Connect client, IMS TM Resource Adapter.

IMS Connect Event Recorder exit routine (HWSTECL0)

IMS Connect can be customized to facilitate event recording by passing event data to the load module, HWSTECL0. HWSTECL0 stores all trace and event notifications through a recording routine and can be used by any event recording function.

For performance or basic data analysis, you can record events such as:

- TCP/IP read/write
- RACF calls
- OTMA sends and receives
- User exit calls
- Session errors

- Two-phase commit events
- Connection and message events for IMS-to-IMS TCP/IP communication
- Connection and message events for ISC TCP/IP communication
- Changes to IMS Connect resources that are made by IMS type-2 commands

IMS Connect provides a sample HWSTECL0 user exit for you to customize.

Subsections:

- "HWSTECL0 initialization" on page 690
- "Invoking the HWSTECL0 for user exit event recording" on page 691
- "Error message format" on page 692

HWSTECL0 initialization

When IMS Connect initializes, IMS Connect automatically loads the HWSTECL0 module and calls the module for event recording initialization. If event and trace recording is detected and is active, module HWSTECL0 sets the Event Interface Control Block (EICB) fields, which is used to control event recording, to the appropriate values needed for event and trace recording.

The address of the EICB is pointed to by HWSTECL0 register 1 on entry. Note, event initialization only occurs if the caller is executing under the JOBSTEP TCB, the caller is in primary TCB mode, and the call occurs before any task that records events is created.

The following table describes the registers at entry to HWSTECL0.

Table 324. Registers at entry to HWSTECL0

Register Number	Contents and meaning
R1	Address of the Event Interface Control Block (EICB) that is to be completed by HWSTECL0 when trace or event recording is active.
R13	Address of save area that is a set of pre-chained save areas. HWSTECL0 must preserve the integrity of the save area set.
R14	Caller's return address.
R15	Entry point of module HWSTECL0.

The EICB area is allocated by IMS Connect and passed to HWSTECL0 at the initialization request. The DSECT name is HWSECIB. If trace or event recording is active, HWSTECL0 completes the EICB and returns it to the caller. The contents of the control block that are returned from HWSTECL0 are shown in the following table.

Table 325. Contents of Event Interface Control Block (EICB) pointed to by HWSTECL0

Element	Length	Usage and meaning
EYECATCHER	4	Value of EICB identifying this block in working storage. Set by caller.
FLAGS	1	Interface control flags: 1. Event recording is enabled.
EVENT_TOKEN	4	Address of the token used by the event recording routine. The token must be passed to the event recording routine when an event-recording request is made.
EVENT_ADDRESS	4	Entry address of event recording routine.

Table 325. Contents of Event Interface Control Block (EICB) pointed to by HWSTECL0 (continued)

Element	Length	Usage and meaning
	4	Reserved space.
	4	Reserved space.
MESSAGE_LEN	2	Length of the message returned from HWSTECL0 module.
MESSAGE_AREA	120	An area that can be used by HWSTECL0 to return an informational or error message to IMS Connect.

If trace or event recording is not active, HWSTECL0 does not complete the EICB and instead returns with a return and reason code indicating that trace or event recording, or both is not active. The following table describes the registers at return from HWSTECL0. **Note:** Module HWSTECL0 always returns a return code of 0. The EICB flags must be inspected to determine if event or trace recording is active.

Table 326. Registers at return from HWSTECL0

Register number	Contents and meaning
R0	Reason code associated with any non-zero return codes passed.
R15	Return code • 0 = Initialization was successful. Check the EICB to see if trace or event recording is active. • 8 = Initialization was not successful. See reason code for additional information.

Invoking the HWSTECL0 for user exit event recording

When IMS Connect records an event, IMS Connect calls the event recording routine address, EVENT_ADDRESS, indicated in the EICB. For each event that is recorded, the event recording routine passes the Event Record Parameter List (ERPL), which is used to define the event type and event data. The ERPL defines which event data to capture. The ERPL records an IMS Connect event and associated data to an event-recording log.

When event recording has been initialized, the EICB contains the entry address for event recording and calls the event recording routine. The routine points to the ERPL address and records the event. To record an event, the caller requesting event recording must be in primary TCB mode and the caller must return the event recording token which is provided in the EICB by HWSTECL0.

The following table describes the registers at event recording entry.

Table 327. Registers at event recording entry

Register number	Contents and meaning
R1	Address of the Event recording parameter list (ERPL).
R13	Address of one save area. The event recording routine must preserve the integrity of the save area.
R14	Caller's return address.
R15	Entry point of even recording taken from EICB after initialization of the even recording interface.

The following table shows the registers at return from EICB, the event recording interface.

Table 328. Registers at return from event recording

Register number	Contents and meaning
R0	Reason code associated with any non-zero return codes passed.
R1	When R1 is not equal to zero, it contains the address of a message providing additional information about initialization of trace and event recording.
R15	Return code • 0 = Event recording was successful. • 4 = Event recording is not active -- event was not recorded. • 16 = Event recording was not successful. See reason code for additional information. An error message is present if R1 is not zero.

Error message format

If an error message is returned by the event-recording routine, the format of the error message is described in the following table. Note, the use of error messages is optional and currently is not supported.

Table 329. Error message format

Value	Contents and meaning
2 byte message length	The true length of the error message not including the message length field.
Error message	An error message returned by the recording exit.

Related reference

"Event Interface Control Block (EICB)" on page 755
The EICB links IMS Connect and the trace and event recording module, HWSTECL0.

"Event recording parameter list (ERPL)" on page 754
The ERPL is used to record an IMS Connect event and associated data to an event-recording log.

Modifying the HWSTECL0 user exit

Although IMS Connect provides a sample HWSTECL0 user exit, you must modify the HWSTECL0 user exit, using standard user-exit development guidelines, if you want to receive event data from IMS Connect.

The source code for the HWSTECL0 user exit is located in the ADFSSMPL source library.

After you have customized the sample HWSTECL0 user exit, you must install it into your IMS Connect resource library (SDFSRESL). To install HWSTECL0 into the resource library, you must compile and bind the user exit before you execute IMS Connect to create the load module, HWSTECL0. IMS Connect will load your HWSTECL0 module from the resource library and call it during initialization and termination.

The following steps describe how to customize, modify, and re-install the HWSTECL0 exit.

1. Insert your changes to the source code provided in the ADFSSMPL source library.
2. Assemble the exit. The exit and its associated macro files are members of the partitioned data set into which you receive the ADFSSMPL data set.
3. Bind the output from the assembled job to create a load module named HWSTECL0.
4. Bind HWSTECL0 into the IMS Connect resource library, SDFSRESL. IMS Connect loads the module from the resource library during initialization.

Related reference

"DSECTs for event recording" on page 762

Macros are shipped with IMS Connect to help customize with event recording.

Event types

The IMS Connect Event Recorder exit routine stores and categorizes event notifications using key values, event numbers, and event keys.

An event can be a *single event* or a *multiple event*.

Each event is assigned a numeric value called an event number. Each event also has an associated key value, such as EVNT or SVTOKEN.

An event with an event number of 255 includes a 2-byte extended event number that follows the event number field. For these events, the extended event number identifies the event.

Event keys

The event key value is an identifier of the type of the event.

The key value EVNT indicates a single event. The key value SVTOKEN indicates a multiple event process.

The following table describes the key values and the length of the event key.

Table 330. Keys associated with events

Key value	Length	Usage and meaning
EVNT	8	This is a constant value (EVNT) used to indicate the event is not associated with a multi-event process. The constant is left-justified and padded right with blanks.
SVT token value	8	SVT Token. A token representing the SVT control block for the remote client name associated with the transaction or multi-event process. The token is the STCK time of when the SVT was created.
Session token value	8	Session token. A token representing a sequence of related events. The token is the STCK time of the first event in the sequence
Command token value	8	Command token. The token is the STCK time of when the command was entered to the Operations Manager (OM).

Single process event types

A single process event is an event that is not related to any other events.

The following table identifies the events that are categorized as a single event type. The following table lists the possible single events that may be recorded.

Table 331. Single process events

Event number	Extended event number	Event key	Event description
1		EVNT	Connect region initialization. This event record is generated as a result of the call made to module HWSTECL0 for event recording initialization. This is the first event recorded for an IMS Connect execution.
2		EVNT	Connect region has completed termination. This event is the last event recorded for an IMS Connect execution. This event causes the event recording process to terminate.

Table 331. Single process events (continued)

Event number	Extended event number	Event key	Event description
3		EVNT	A support task (TCB) has been created. If the task records events, this event must be the first event recorded by the task. It should be recorded as soon as possible after the task begins processing.
4		EVNT	A support task (TCB) is terminating. If the task records events, this event must be the last event recorded by the task. It should be recorded as close as possible to the task returning to MVS.
5		EVNT	Begin INIT API.
6		EVNT	End INIT API.
7		EVNT	Begin Bind socket.
8		EVNT	End Bind socket.
9		EVNT	Listen on socket.
10		EVNT	Begin Accept socket.
11		EVNT	End Accept socket.
Note			Events 12 and 13 are defined in the section on multi-event types.
14		EVNT	Begin init of message exits. This event serves to initialize the task for message exit processing.
16		EVNT	IMS data store becomes available. The event is recorded during the following processes. It represents a successful client bid process. 1. During IMS Connect initialization - once for each available data store. 2. After IMS Connect initialization - any time a data store joins the z/OS cross-system coupling facility group and client bid is completed.
17		EVNT	IMS data store becomes unavailable. This event represents a data store that has become unavailable for transactions. It could be due to a stop data store command or the data store member leaving the XCF group. The event is recorded for either occurrence.
18		EVNT	An IMS TMEMBER joins the XCF group.
19		EVNT	An IMS TMEMBER leaves the XCF group.
20		EVNT	Begin SCI registration.
21		EVNT	End SCI registration.
22		EVNT	Begin SCI De-registration.
23		EVNT	End SCI De-registration.

Table 331. Single process events (continued)

Event number	Extended event number	Event key	Event description
24		EVNT	Recorded trace DCB has been opened. This event recorded after the recorder trace DCB has been successfully opened.
25		EVNT	Recorded trace DCB pre-close. This event is recorded when the recorder trace DCB is about to be closed. This event is recorded while the recorder trace DCB is still open.
26		EVNT	User message exit return from INIT. This event is recorded just after the user message exit returns.
27		EVNT	User message exit return from TERM. This event is recorded just after the user message exit returns.
28		EVNT	Begin Secure Environment Open. This is issued at the start of SSL environment creation.
29		EVNT	End Secure Environment Open. This is issued at the end of SSL environment creation.
32		EVNT	Begin Secure Environment Close. This is issued at the start of SSL close.
33		EVNT	End Secure Environment Close. This is issued at the end of SSL initialization.
34		EVNT	Begin Local Port Setup. This event is recorded when a local port is present.
35		EVNT	End Local Port Setup. This event is recorded when a local port is present.
36		EVNT	Begin z/OS Resource Recovery Services Connect. This event is recorded when RRS connect processing is started.
37		EVNT	End RRS Connect. This event is recorded when RRS connect processing is completed.
38		EVNT	List In-doubt Context. This event records the receipt of an in-doubt context during RRS connect processing.
39		EVNT	Begin RRS Disconnect. This event is recorded when RRS disconnect processing is started.
40		EVNT	End RRS Disconnect. This event is recorded when RRS disconnect processing is completed.
41		EVNT	ODBM registration begin.
42		EVNT	ODBM registration end.
43		EVNT	ODBM de-registration begin.
44		EVNT	ODBM de-registration end.

Table 331. Single process events (continued)

Event number	Extended event number	Event key	Event description
45		EVNT	This event is recorded when the Exit Interface Block Data Store (XIBDS) identifies OTMA resources that are in a severe, warning, or normal condition.
46		EVNT	IMS Connect Port Message Edit exit routine initialization.
47		EVNT	IMS Connect Port Message Edit exit routine termination.
48		EVNT	Begin IMS Connect ODBM Routing exit routine initialization.
49		EVNT	End IMS Connect ODBM Routing exit routine initialization.
50		EVNT	Begin IMS Connect ODBM Routing exit routine termination.
51		EVNT	End IMS Connect ODBM Routing exit routine termination.
52		EVNT	XML Adapter INIT call begin. This event is recorded just prior to calling the XML Adapter INIT function.
53		EVNT	XML Adapter INIT call end. This event is recorded just after the XML Adapter INIT function returns.
54		EVNT	XML Adapter TERM call begin. This event is recorded just prior to calling the XML Adapter TERM function.
55		EVNT	XML Adapter TERM call end. This event is recorded just after the XML Adapter TERM function returns
56		EVNT	OM registration.
57		EVNT	OM deregistration.
113		EVNT	Connected to remote IMS Connect socket.
114		EVNT	Disconnected from remote IMS Connect socket.
115		EVNT	Communications thread started for a remote IMS Connect connection.
124		EVNT	Connection to remote IMS Connect timed out.
255	256	EVNT	Socket connected on RMTCICS.
255	257	EVNT	Socket disconnected from RMTCICS.
255	258	EVNT	IMS Connect refreshed a cached RACF user ID after receiving a type 71 Event Notification Facility (ENF) notification.
255	259	EVNT	IMS Connect sent a health status report to Work Load Manager (WLM).
255	771	EVNT	PORT resource created.

Table 331. Single process events (continued)

Event number	Extended event number	Event key	Event description
255	772	EVNT	PORT resource deleted.
255	773	EVNT	PORT resource updated.
255	774	EVNT	DATASTORE resource created.
255	775	EVNT	DATASTORE resource deleted.
255	776	EVNT	DATASTORE resource updated.
255	777	EVNT	IMSPLEX resource created.
255	778	EVNT	IMSPLEX resource deleted.
255	2050	EVNT	Communication thread started for a RMTCICS connection.

Multiple process event types

A multiple event is a series of events that are closely related to each other within a process such as a transaction.

The following table identifies the events that are categorized as a multiple event type. The following table lists the possible multiple events that can be recorded.

Table 332. Multi-process events

Event number	Extended event number	Event key	Event description
12		SVT Token	Begin close socket.
13		SVT Token	End close socket.
60		SVT Token	Prepare for socket read. This is the start-of-frame event for a multi-event process. It is the first event associated with an SVT Token.
61		SVT Token	User message exit entered for READ, XMIT, or EXER. This event is recorded just prior to calling the user message exit.
62		SVT Token	User message exit return for READ, XMIT, or EXER. This event is recorded just after the user message exit returns.
63		SVT Token	Begin SAF security request.
64		SVT Token	End SAF security request.
65		SVT Token	Message sent to OTMA. This entry is made after the message has been sent to OTMA.
66		SVT Token	Message received from OTMA. This entry is made when a message has been received from OTMA. It is recorded after all parts of the message have been assembled.
67		SVT Token or command token	Message sent to SCI. TYPE=CMDINPUT or TYP2RESP.

Table 332. Multi-process events (continued)

Event number	Extended event number	Event key	Event description
68		SVT Token or command token	Message received from SCI. TYPE=CMDRESP or TYP2INPT.
69		SVT Token	OTMA timeout. This event signals that a timeout occurred for an OTMA request.
70		SVT Token	De-allocate request. This event is generated when IMS Connect honors a request from the remote client to disconnect the session.
71		SVT Token	Session error. This event is called when an unrecoverable error has been encountered and the session is being aborted. For this error condition, this should probably be the last event before the trigger event is recorded.
72		SVT Token	Trigger event. This is the end-of-frame event recorded by IMS Connect when a multi-event process has completed.
73		SVT Token	Read socket.
74		SVT Token	Write socket.
80		SVT Token	Begin create context. This event records the request to RRS to create a context for a transaction requesting two-phase commit support.
81		SVT Token	End create context. This event records the end of creation of context for a transaction requesting two-phase commit support.
82		SVT Token	Begin RRS prepare. This event records sending the prepare-to-commit request to RRS.
83		SVT Token	End RRS prepare. This event records receiving the response to the prepare-to-commit request.
84		SVT Token	Begin RRS commit/abort. This event records sending the commit/abort request to RRS.
85		SVT Token	End RRS commit/abort. This event records receiving a response to the commit/abort request.
86		SVT Token	Begin secure environment select. This is issued at the end of SSL select.
87		SVT Token	End secure environment select. This is issued at the end of SSL select.
88		SVT Token	Entire message received from the OTMA asynchronous tpipe hold queue in response to a RESUME TPIPE call. This event is recorded at the end of message assembly.
89		SVT Token	IMS Connect Port Message Edit exit routine entered. This event is recorded just prior to calling the exit.

Table 332. Multi-process events (continued)

Event number	Extended event number	Event key	Event description
90		SVT Token	IMS Connect Port Message Edit exit routine entered. This event is recorded just after the exit returns.
91		SVT Token	DRDA distributed data management (DDM) command.
92		SVT Token	DRDA DDM reply.
93		SVT Token	APSB begin.
94		SVT Token	APSB end.
95		SVT Token	DPSB begin.
96		SVT Token	DPSB end.
97		SVT Token	Enter routing exit.
98		SVT Token	Return from routing exit routine.
99		SVT Token	Enter security exit.
100		SVT Token	Return from security exit routine.
101		SVT Token	RRS PUR begin.
102		SVT Token	RRS PUR end.
103		SVT Token	RRS SWID begin.
104		SVT Token	RRS SWID end.
105		SVT Token	Message sent to ODBM.
106		SVT Token	Message received from ODBM.
107		SVT Token	RRS Delegate Commit Agent UR begins. This event is recorded just prior to calling RRS.
108		SVT Token	RRS Delegate Commit Agent UR ends. This event is recorded just after returning from RRS.
109		SVT Token	XML Adapter RXML or XXML call begin. This event is recorded just prior to calling the XML Adapter RXML or XXML function.
110		SVT Token	XML Adapter RXML or XXML call end. This event is recorded just after the XML Adapter RXML or XXML function returns.
111		SVT Token	XML converter call begin. This event is recorded just prior to calling the XML converter.
112		SVT Token	XML converter call end. This event is recorded just after the XML converter returns.
116		Session token	Message received from OTMA for OTMA remote ALTPCB function. TYPE=REQUEST or ACK/NAK.
117		Session token	Message sent to remote IMS Connect over TCP/IP for OTMA remote ALTPCB function. TYPE=REQUEST or ACK/NAK.

Table 332. Multi-process events (continued)

Event number	Extended event number	Event key	Event description
118		Session token	Message received from remote IMS Connect over TCP/IP for OTMA remote ALTPCB function. TYPE=REQUEST or ACK/NAK.
119		Session token	Message sent to OTMA for OTMA remote ALTPCB function. TYPE=REQUEST or ACK/NAK.
120		Session token	MSC message received from MSC. TYPE=REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, or SHUTDDIR.
121		Session token	MSC message sent to remote IMS Connect. TYPE=REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, SHUTDDIR, or ERRORRSP.
122		Session token	MSC message received from remote IMS Connect. TYPE=REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, SHUTDDIR, or ERRORRSP.
123		Session token	MSC message sent to MSC. TYPE=REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, SHUTDDIR, or ERRORRSP.
125		Session token	Start of a session.
126		Session token	Trigger for the end of a session.
255	2051	Session token	ISC message received from IMS.
255	2052	Session token	ISC message sent to IMS.
255	2053	Session token	ISC message received on RMTCICS socket connection.
255	2054	Session token	ISC message sent on RMTCICS socket connection.
255	2055	Session token	ISC message received on CICSPORT socket connection.
255	2056	Session token	ISC message sent on CICSPORT socket connection.

Event record formats

The IMS Connect Event Recorder exit routine stores and categorizes the event and format for all event records using the ERPL (Event Record Parameter List).

The following tables list the format for all event records. Each table identifies each possible event in the ERPL that can be recorded to the HWSTECL0 module and provides the format for each event.

The following table identifies the parameter list content associated with the IMS Connect region initialization event.

Table 333. Connect region initialization event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	1	2

Table 333. Connect region initialization event (continued)

Parameter list item	Content	Length in bytes
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	4	2
VAR_DATA	Start of variable data area.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_VVRR	IMS Connect Version and Release data.	2

The following table identifies the parameter list contents associated with the Connect region termination.

Table 334. Connect region termination event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	2	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of variable data area.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_COMPCODE	Completion code associated with region termination.	4

The following table identifies the parameter list contents associated with the Support Task Created event.

Table 335. Support task created event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	3	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of variable data area.	0
VAR_APAR	APAR sequence number for the control block.	2

Table 335. Support task created event (continued)

Parameter list item	Content	Length in bytes
VAR_FLAG	Flag field indicating TCB type: • x'80' port task • x'40' local port task • x'20' recorder task • x'10' dynamically added • x'08' DRDA port • Reserved	1
VAR_PORT	Port number if port task.	2
VAR_EDITXT	Port edit exit if dynamically added.	8
VAR_KEEPAV	Port keep alive if dynamically added.	4
VAR_PORTTO	DRDA port timeout if dynamically added.	4
VAR_IDLETO	Port idle timeout.	4

The following table identifies the parameter list contents associated with the Support Task Terminating event.

Table 336. Support task terminating event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	4	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_FLAG	Flag field indicating TCB type: 1. port 2. local 3. recorder	2
VAR_PORT	Port number if port task.	2

The following table identifies the parameter list contents associated with the event, Begin Initialize API.

Table 337. Begin initialize API event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	5	2
	Reserved	2

Table 337. Begin initialize API event (continued)

Parameter list item	Content	Length in bytes
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the event, End Initialize API.

Table 338. End initialize API event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	6	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	10	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Begin Bind Socket event. If this is a secure socket (SSL), the TCPIB (TCP/IP Information Block) contains a flag indicating the operation is executing against an SSL port.

Table 339. Begin bind socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	7	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with the End Bind Socket event. If this is a secure socket (SSL), the TCPIB (TCP/IP Information Block) contains a flag indicating the operation is executing against an SSL port.

Table 340. End bind socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4

Table 340. End bind socket event (continued)

Parameter list item	Content	Length in bytes
EVENT_NUMBER	8	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB.	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Listen on Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operations is executing against an SSL port.

Table 341. Listen on socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	9	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with the Begin Accept Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operation is executing against an SSL port.

Table 342. Begin accept socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	10	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the End Accept Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operation is executing against an SSL port.

Table 343. End accept socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	11	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB.	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Begin Close Socket event. If this is a secure socket (SSL), the TCPIB (TCP/IP Information Block) contains a flag indicating that the operations is executing against an SSL port.

Table 344. Begin close socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	12	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the End Close Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operations is executing against an SSL port.

Table 345. End close socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	13	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2

Table 345. End close socket event (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB.	4
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list content associated with the Begin Initialization of Message Exits event.

Table 346. Begin initialization of message exits

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	14	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the Data Store Available event.

Table 347. Data Store available event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	16	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the Data Store Unavailable event.

Table 348. Data Store unavailable event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	17	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2

Table 348. Data Store unavailable event (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the TMEMBER Joins z/OS cross-system coupling facility Group event.

Table 349. TMEMBER joins XCF group event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	18	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the TMEMBER Leaves XCF Group event.

Table 350. TMEMBER leaves XCF group event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	19	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the Begin SCI Registration event.

Table 351. Begin SCI registration event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	20	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2

Table 351. Begin SCI registration event (continued)

Parameter list item	Content	Length in bytes
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the End SCI Registration event.

Table 352. End SCI registration event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	21	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB.	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Begin SCI De-registration event.

Table 353. Begin SCI de-registration event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	22	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the End SCI De-registration event.

Table 354. End SCI de-registration event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	23	2
	Reserved	2
EVENT_KEY	EVNT	8

Table 354. End SCI de-registration event (continued)

Parameter list item	Content	Length in bytes
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB.	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Recorder Trace DCB Opened event.

Table 355. Recorder trace DCB opened event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	24	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the recorder trace DCB.	4

The following table identifies the parameter list contents associated with the Recorder Trace DCB Pre-close event.

Table 356. Recorder trace DCB pre-close event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	25	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the Message Exit INIT Call event.

Table 357. Message exit INIT call event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	26	2

Table 357. Message exit INIT call event (continued)

Parameter list item	Content	Length in bytes
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	18	2
EVENT_DATA_ADDR	Address of the exit parameter list.	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4
VAR_EXIT_NAME	Name of the user message exit.	8

The following table identifies the parameter list contents associated with the Message Exit TERM Call event.

Table 358. Message exit TERM call event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	27	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4
VAR_EXIT_NAME	Name of the user message exit.	8

The following table identifies the parameter list contents associated with the Begin Secure Environment Open event.

Table 359. Begin secure environment open event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	28	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2

Table 359. Begin secure environment open event (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the End Secure Environment Open event.

Table 360. End secure environment open event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	29	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB.	
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Begin Secure Environment Close event.

Table 361. Begin secure environment close event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	32	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the End Secure Environment Close event.

Table 362. End secure environment close event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	33	2

Table 362. End secure environment close event (continued)

Parameter list item	Content	Length in bytes
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Begin Local Port Setup event.

Table 363. Begin local port setup event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	34	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the End Local Port Setup event.

Table 364. End local port setup event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	35	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	14	2
EVENT_DATA_ADDR	Address of the TCPIB.	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Begin z/OS Resource Recovery Services Connect event.

Table 365. Begin RRS connect event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	36	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the End RRS Connect event.

Table 366. End RRS connect event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	37	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4

The following table identifies the parameter list contents associated with the List In-doubt Context event.

Table 367. List in-doubt context event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	38	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	162	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_URTOKEN	The UR_INTEREST_TOKEN returned by RRS.	16
VAR_XID	The XID associated with this transaction	140

The following table identifies the parameter list contents associated with the Begin RRS Disconnect event.

Table 368. Begin RRS disconnect event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	39	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the End RRS Disconnect event.

Table 369. End RRS disconnect event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	40	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4

The following table identifies the parameter list contents associated with the ODBM registration begin event.

Table 370. ODBM registration begin event 41

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	41	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB	4

The following table identifies the parameter list contents associated with the ODBM registration end event.

Table 371. ODBM registration end event 42

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	42	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
EVENT_DATA_ADDR	Address of DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code.	4
VAR_RSNCODE	Reason code	4

The following table identifies the parameter list contents associated with the ODBM de-registration begin event.

Table 372. ODBM de-registration begin event 43

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	43	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB	4

The following table identifies the parameter list contents associated with the ODBM de-registration end event.

Table 373. ODBM de-registration end event 44

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	44	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2

Table 373. ODBM de-registration end event 44 (continued)

Parameter list item	Content	Length in bytes
VAR_RETCODE	Return code	4
VAR_RSNCODE	Reason code	4

The following table identifies the parameter list contents associated with the Exit Interface Block Data Store (XIBDS) status update event.

Table 374. XIBDS status update event 45

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	45	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the XIBDS	4

The following table identifies the parameter list contents associated with the Port Edit exit INIT event.

Table 375. Port Edit exit INIT event 46

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	46	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_RC	Return code from exit	4
VAR_RSN	Reason code form exit	4
VAR_EXITN	Name of User Exit	8

The following table identifies the parameter list contents associated with the Port Edit exit TERM event.

Table 376. Port Edit exit TERM event 47

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	47	2
	Reserved	2

Table 376. Port Edit exit TERM event 47 (continued)

Parameter list item	Content	Length in bytes
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_RC	Return code from exit	4
VAR_RSN	Reason code form exit	4
VAR_EXITN	Name of User Exit	8

The following table identifies the parameter list contents associated with the begin IMS Connect ODBM routing exit routine initialization event.

Table 377. IMS Connect ODBM Routing exit routine initialization begin event 48

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	48	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_EXITNAME	Exit routine name	8

The following table identifies the parameter list contents associated with the end IMS Connect ODBM routing exit routine initialization event.

Table 378. IMS Connect ODBM Routing exit routine initialization end event 49

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	49	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	18	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data.	0

Table 378. IMS Connect ODBM Routing exit routine initialization end event 49 (continued)

Parameter list item	Content	Length in bytes
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code	4
VAR_RSNCODE	Reason code	4
VAR_EXITNAME	Exit routine name	8

The following table identifies the parameter list contents associated with the begin IMS Connect ODBM routing exit routine termination event.

Table 379. IMS Connect ODBM Routing exit routine termination end event 50

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	50	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_EXITNAME	Exit routine name	8

The following table identifies the parameter list contents associated with the end IMS Connect ODBM routing exit routine termination event.

Table 380. IMS Connect ODBM Routing exit routine termination end event 51

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	51	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	18	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code	4
VAR_RSNCODE	Reason code	4
VAR_EXITNAME	Exit routine name	8

The following table identifies the parameter list content associated with the XML Adapter INIT call begin event.

Table 381. XML Adapter INIT call begin event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	52	4
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	10	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_ADAPTER_NAME	Adapter name	8

The following table identifies the parameter list content associated with the XML Adapter INIT call end event.

Table 382. XML Adapter INIT call end event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	53	4
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_RC	Return code	4
VAR_RSN	Reason code	4
VAR_ADAPTER_NAME	Adapter name	8

The following table identifies the parameter list content associated with the XML Adapter TERM call begin event.

Table 383. XML Adapter TERM call begin event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	54	4
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	10	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2

Table 383. XML Adapter TERM call begin event (continued)

Parameter list item	Content	Length in bytes
VAR_ADAPTER_NAME	Adapter name	8

The following table identifies the parameter list content associated with the XML Adapter TERM call end event.

Table 384. XML Adapter TERM call end event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	55	4
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_RC	Return code	4
VAR_RSN	Reason code	4
VAR_ADAPTER_NAME	Adapter name	8

The following table identifies the parameter list content associated with the OM registration event.

Table 385. OM registration event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	56	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	20	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
	Reserved	2
VAR_RC	Return code	4
VAR_RSN	Reason code	4
VAR_OM_NAME	Name of the OM	8

The following table identifies the parameter list content associated with the OM deregistration event.

Table 386. OM deregistration event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	57	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	12	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
	Reserved	2
VAR_RC	Return code	4
VAR_RSN	Reason code	4

The following table identifies the parameter list contents associated with the Prepare Socket Read event.

Table 387. Prepare socket read event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	60	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the Message Exit Called for READ, XMIT, or EXER event.

Table 388. Message exit called for READ, XMIT, or EXER event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	61	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	2	2
VAR_DATA_LL	18	2
EVENT_DATA_ADDR	Address of the parameter list at entry (R1).	4

Table 388. Message exit called for READ, XMIT, or EXER event (continued)

Parameter list item	Content	Length in bytes
EVENT_DATA_ADDR2	If READ or EXER, address of the IRM (IMS request message) header. If XMIT, address of the OTMA header.	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_EXIT_NAME	Exit name.	8
VAR_TRACKINGID_ADDRS	Address of the tracking ID on output.	4
VAR_TRACKINGID_LEN	Length of the tracking ID on output.	4

The following table identifies the parameter list contents associated with the Message Exit Return for READ, XMIT, or EXER event.

Table 389. Message exit return for READ, XMIT, or EXER event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	62	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	2	2
VAR_DATA_LL	26	2
EVENT_DATA_ADDR	Address of the parameter list at entry (R1).	4
EVENT_DATA_ADDR2	If XMIT or EXER, address of the remote client message. If READ, address of the OTMA header.	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4
VAR_EXIT_NAME	Exit name.	8
VAR_TRACKINGID_ADDRS	Address of the tracking ID on output.	4
VAR_TRACKINGID_LEN	Length of the tracking ID on output.	4

The following table identifies the parameter list contents associated with the Begin SAF Request event.

Table 390. Begin SAF request event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	63	2
	Reserved	2
EVENT_KEY	SVT Token	8

Table 390. Begin SAF request event (continued)

Parameter list item	Content	Length in bytes
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the SAFIB.	4

The following table identifies the parameter list contents associated with the End SAF Request event.

Table 391. End SAF request event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	64	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the SAFIB.	4

The following table identifies the parameter list contents associated with the Message Sent to OTMA event.

Table 392. Message sent to OTMA event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	65	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with the Message Received from OTMA event.

Table 393. Message received from OTMA event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	66	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2

Table 393. Message received from OTMA event (continued)

Parameter list item	Content	Length in bytes
EVENT_DATA_ADDR	Address of the DSIB.	4

The following table identifies the parameter list contents associated with a Message Sent to SCI event.

Table 394. Message sent to SCI event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	67	2
	Reserved	2
EVENT_KEY	**SVT token** Client-initiated command input **Command token** SPOC/OM-initiated type-2 command reply	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB.	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	CMDINPT or TYP2RESP	8

The following table identifies the parameter list contents associated with a Message Received from SCI event.

Table 395. Message received from SCI event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	68	2
	Reserved	2
EVENT_KEY	**SVT token** Client-initiated command reply **Command token** SPOC/OM-initiated type-2 command input	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB.	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	CMDRESP or TYP2INPT	8

The following table identifies the parameter list contents associated with an OTMA timeout event.

Table 396. OTMA timeout event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	69	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_TO_VALUE	Timeout value.	4

The following table identifies the parameter list contents associated with a De-allocate Session event.

Table 397. De-allocate session event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	70	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_DEALC_RSN	Reason for session de-allocation. Note: Can be a flag or constant type of reason.	4

The following table identifies the parameter list contents associated with a Session Error event.

Table 398. Session error event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	71	2
	Reserved	2
EVENT_KEY	SVT Token or EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	154	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2

Table 398. Session error event (continued)

Parameter list item	Content	Length in bytes
VAR_FLAG	Flag fields indicating record content. 1. Message is present in the record. 2. Out-of-frame error.	2
VAR_MESSAGE	If a message is generated for the error, it is contained in this field.	134
VAR_SESS_RSN	Reason for the session de-allocation. Note: The session reason is a character expression of the error type.	8
VAR_SESS_TOKEN	The SVTTOKEN associated with the message when the session error occurs out-of-frame and the SVTTOKEN for the message cannot be located by IMS Connect. Note: This field is valid only when the key of the event is EVNT. If the key is an SVTTOKEN value, this field is zero. In some cases, where asynchronous output is created by a non-IMS Connect source, the field may contain values that do not resemble a normal IMS Connect SVTTOKEN.	8

The following table identifies the parameter list contents associated with a Trigger event.

Table 399. Trigger event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	72	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	10	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_TRIG_TYPE	Constant identifying triggers type. Values can be TRAN or TPIPE or anything else that is needed.	8

The following table identifies the parameter list contents associated with a Read Socket event.

Table 400. Read socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	73	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2

Table 400. Read socket event (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the Write Socket event.

Table 401. Write socket event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	74	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB.	4

The following table identifies the parameter list contents associated with the Begin Create Context event.

Table 402. Begin create context event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	80	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the End Create Context event.

Table 403. End create context event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	81	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	162	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	RRS return code.	4

Table 403. End create context event (continued)

Parameter list item	Content	Length in bytes
VAR_URTOKEN	UR Interest token returned from RRS.	16
VAR_XID	The remote client XID associated with the transaction.	140

The following table identifies the parameter list contents associated with the Begin RRS Prepare event.

Table 404. Begin RRS prepare event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	82	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_URTOKEN	URTOKEN associated with the request.	16

The following table identifies the parameter list contents associated with the End RRS Prepare event.

Table 405. End RRS prepare event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	83	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	24	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_RC	Return code.	4
VAR_FLAG	Result flags: 1. At least 1 participant replied abort. Note: The results flag is set if any participant has requested the context be aborted.	2
VAR_URTOKEN	URTOKEN associated with the request.	16

The following table identifies the parameter list contents associated with the Begin RRS Commit/Abort event.

Table 406. Begin RRS commit/abort event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	84	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	20	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_FLAG	Result flags: 1. request to abort 2. request to commit Note: The results flag is set if any participant has requested the context be aborted.	2
VAR_URTOKEN	URTOKEN associated with the request.	16

The following table identifies the parameter list contents associated with the End RRS Commit/Abort event.

Table 407. End RRS commit/abort event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	85	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	24	2
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_FLAG	Result flags: 1. request to abort 2. request to commit 3. could not find the URTOKEN Note: The results flag is set if any participant has requested the context be aborted.	2
VAR_RC	Return code.	4
VAR_URTOKEN	URTOKEN associated with the request.	16

The following table identifies the parameter list contents associated with the Begin Secure Environment Select event.

Table 408. Begin secure environment select event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	86	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	4	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_FLAG	Result flags: 1. Select for Read. 2. Select for Write	2

The following table identifies the parameter list contents associated with the End Secure Environment Select event.

Table 409. End secure environment select event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	87	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	12	2
EVENT_DATA_ADDR	Address of the TCPIB.	4
VAR_DATA	Start of the variable data.	0
VAR_APAR	APAR sequence number for the control block.	2
VAR_FLAG	Result flags: 1. Select for Read. 2. Select for Writer.	2
VAR_RC	Return code.	4
VAR_RSN	Reason code.	4

The following table identifies the parameter list contents associated with the Message Received from OTMA event by the Resume Tpipe call.

Table 410. Message received from OTMA event by the Resume Tpipe call

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	88	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	2	2
VAR_DATA_LL	8	2
EVENT_DATA_ADDR1	Address of the parameter list at entry	4
EVENT_DATA_ADDR2	Address of the SVT TOKEN of INPUT SVI	4
VAR_DATA	Start of variable data	0
VAT_EXIT_NAME	Exit name	8

The following table identifies the parameter list contents just before the IMS Connect Port Message Edit exit routine call.

Table 411. Port Edit exit begin event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	89	2
	Reserved	2
EVENT_KEY	SVT TOKEN	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	14	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_PARML	HWSEXPIO parameter list address	4
VAR_EXITN	Exit name	8

The following table identifies the parameter list contents associated with the return from the IMS Connect Port Message Edit exit routine call.

Table 412. Port Edit exit return event

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	90	2
	Reserved	2
EVENT_KEY	SVT TOKEN	8
DATA_ADDR_COUNT	2	2
VAR_DATA_LL	8	2

Table 412. Port Edit exit return event (continued)

Parameter list item	Content	Length in bytes
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_PARML	HWSEXPIO parameter list address	4
VAR_RC	Return code	4
VAR_RSN	Reason code	4
VAR_EXITN	Name of User Exit	8

The following table identifies the parameter list contents associated with the DRDA distributed data management (DDM) command event.

Table 413. DRDA DDM command event 91

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	91	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	4	2
EVENT_DATA_ADDR	Address of the DRDA command	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_CODEPOINT	DRDA DDM command codepoint	2

The following table identifies the parameter list contents associated with the DRDA DDM command reply event.

Table 414. DRDA DDM command reply event 92

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	92	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	4	2
EVENT_DATA_ADDR	Address of the DRDA DDM reply	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_CODEPOINT	DRDA DDM reply codepoint	2

The following table identifies the parameter list contents associated with the APSB Begin event.

Table 415. APSB Begin event 93

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	93	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	30	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_PSBNAME	PSB name	8
VAR_ALIAS	IMS alias name	4
VAR_STCKE	Store clock	16

The following table identifies the parameter list contents associated with the APSB end event.

Table 416. APSB end event 94

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	94	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	28	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_PSBNAME	PSB name	8
VAR_CODEPOINT	Codepoint	2
VAR_STCKE	Store clock	16

The following table identifies the parameter list contents associated with the DPSB begin event.

Table 417. DPSB begin event 95

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	95	2
	Reserved	2
EVENT_KEY	SVT Token	8

Table 417. DPSB begin event 95 (continued)

Parameter list item	Content	Length in bytes
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	26	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_PSBNAME	PSB name	8
VAR_STCKE	Store clock	16

The following table identifies the parameter list contents associated with the DPSB end event.

Table 418. DPSB end event 96

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	96	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	28	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_PSBNAME	PSB name	8
VAR_CODEPOINT	Codepoint	2
VAR_STCKE	Store clock	16

The following table identifies the parameter list contents associated with the enter routing exit event.

Table 419. Enter routing exit event 97

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	97	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	14	2
EVENT_DATA_ADDR	Address of the parameter list at entry (R1)	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_ALIAS	Pre-selected IMS alias name	4
VAR_CLID	Client ID	8

The following table identifies the parameter list contents associated with the return from routing exit event.

Table 420. Return from routing exit event 98

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	98	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	34	2
EVENT_DATA_ADDR	Address of the parameter list at entry (R1)	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code from the exit	4
VAR_RSNCODE	Reason code from the exit	4
VAR_ALIAS	IMS alias name returned from the exit	4
VAR_ODBMNAME	ODBM name returned from the exit	8
VAR_SERVRTN	Service return code	4
VAR_SERVRSN	Service reason code	8

The following table identifies the parameter list contents associated with the enter security exit event.

Table 421. Enter security exit event 99

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	99	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR1	Address of the parameter list at entry (R1)	4

The following table identifies the parameter list contents associated with the return from security exit event.

Table 422. Return from security exit event 100

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	100	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2

Table 422. Return from security exit event 100 (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	22	2
EVENT_DATA_ADDR	Address of the parameter list at entry (R1)	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for the control block	2
VAR_RETCODE	Return code from the exit	4
VAR_RSNCODE	Reason code from the exit	4
VAR_SERVRTN	Service return code	4
VAR_SERVRSN	Service reason code	8

The following table identifies the parameter list contents associated with the RRS parent UR begin event.

Table 423. RRS parent UR begin event 101

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	101	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	142	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_XID	The XID for creating the parent UR	140

The following table identifies the parameter list contents associated with the RRS parent UR end event.

Table 424. RRS parent UR end event 102

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	102	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	162	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code	4
VAR_PURTOKEN	Parent UR token returned	16

Table 424. RRS parent UR end event 102 (continued)

Parameter list item	Content	Length in bytes
VAR_XID	The XID associated with the parent UR	140

The following table identifies the parameter list contents associated with the RRS SWID begin event.

Table 425. RRS SWID begin event 103

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	103	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	158	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_PURTOKEN	Parent UR token returned	16
VAR_XID	The XID associated with the parent UR	140

The following table identifies the parameter list contents associated with the RRS SWID end event.

Table 426. RRS SWID end event 104

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	104	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	162	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code	4
VAR_PURTOKEN	Parent UR token	16
VAR_XID	The XID associated with the parent UR	140

The following table identifies the parameter list contents associated with the message sent to ODBM event.

Table 427. Message sent to ODBM event 105

Parameter list item	Content	Length in bytes
TOKEN	Token address	4

Table 427. Message sent to ODBM event 105 (continued)

Parameter list item	Content	Length in bytes
EVENT_NUMBER	105	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB	4

The following table identifies the parameter list contents associated with the message received from ODBM event.

Table 428. Message received from ODBM event 106

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	106	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the DSIB	4

The following table identifies the parameter list contents associated with the begin RRS delegate commit event.

Table 429. RRS delegate commit begin event 107

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	107	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	158	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_PURTOKEN	Parent UR token	16
VAR_XID	The XID associated with the parent UR	140

The following table identifies the parameter list contents associated with the end RRS delegate commit event.

Table 430. RRS delegate commit end event 108

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	108	2
	Reserved	2
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	162	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_RETCODE	Return code	4
VAR_PURTOKEN	Parent UR token	16
VAR_XID	The XID associated with the parent UR.	140

The following table identifies the parameter list content associated with the XML Adapter RXML and XXML call begin event.

Table 431. XML Adapter RXML and XXML call begin event 109

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	109	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	22	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_ADAPTER_NAME	Adapter name	8
VAR_ADAPTER_FUNC	Adapter function (RXML or XXML)	4
VAR_CONV_NAME	Converter name	8

The following table identifies the parameter list content associated with the XML Adapter RXML and XXML call end event.

Table 432. XML Adapter RXML and XXML call end event 110

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	110	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	30	2

Table 432. XML Adapter RXML and XXML call end event 110 (continued)

Parameter list item	Content	Length in bytes
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_RC	Return code	4
VAR_RSN	Reason code	4
VAR_ADAPTER_NAME	Adapter name	8
VAR_ADAPTER_FUNC	Adapter function (RXML or XXML)	4
VAR_CONV_NAME	Converter name	8

The following table identifies the parameter list content associated with the XML converter call begin event.

Table 433. XML converter call begin event 111

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	111	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	10	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_CONV_NAME	Converter name	8

The following table identifies the parameter list content associated with the XML converter call end event.

Table 434. XML converter call end event 112

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	112	4
EVENT_KEY	SVT Token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	18	2
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_RC	Return code	4
VAR_RSN	Reason code	4
VAR_CONV_NAME	Converter name	8

The following table identifies the parameter list contents associated with the Connected to remote IMS Connect event.

Table 435. Connected to remote IMS Connect event 113

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	113	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with the Disconnected from remote IMS Connect event.

Table 436. Disconnected from remote IMS Connect event 114

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	114	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	26	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block.	2
VAR_TMEMBER	TMEMBER name if the connection is for OTMA, otherwise this field contains blanks.	8
VAR_LCLPLKID	MSC LCLPLKID name if the connection is for MSC, otherwise this field contains blanks.	8
VAR_LINK	LINK name if the connection is for MSC, otherwise this field contains blanks.	8

The following table identifies the parameter list contents associated with the Communications thread started for a remote IMS Connect connection event.

Table 437. Communications thread started for a remote IMS Connect connection event 115

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	115	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2

Table 437. Communications thread started for a remote IMS Connect connection event 115 (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number for control block	2
VAR_RMTICON	RMTIMSCON name	8

The following table identifies the parameter list contents associated with the Message received from OTMA for OTMA remote ALTPCB function event.

Table 438. Message received from OTMA for OTMA remote ALTPCB function event 116

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	116	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST or ACK/NACK	8

The following table identifies the parameter list contents associated with the Message sent to remote IMS Connect over TCP/IP for OTMA remote ALTPCB function event.

Table 439. Message sent to remote IMS Connect over TCP/IP for OTMA remote ALTPCB function event 117

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	117	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	26	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST or ACK/NACK	8
VAR_TMEMBER	TMEMBER name	8

Table 439. Message sent to remote IMS Connect over TCP/IP for OTMA remote ALTPCB function event 117 (continued)

Parameter list item	Content	Length in bytes
VAR_TPIPE	TPIPE name	8

The following table identifies the parameter list contents associated with the Message received from remote IMS Connect over TCP/IP for OTMA remote ALTPCB function event.

Table 440. Message received from remote IMS Connect over TCP/IP for OTMA remote ALTPCB function event 118

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	118	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	26	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST or ACK/NACK	8
VAR_TMEMBER	TMEMBER name	8
VAR_TPIPE	TPIPE name	8

The following table identifies the parameter list contents associated with the Message sent to OTMA for an OTMA remote ALTPCB function event.

Table 441. Message sent to OTMA for OTMA remote ALTPCB function event 119

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	119	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST or ACK/NACK	8

The following table identifies the parameter list contents associated with the MSC message received from MSC event.

Table 442. MSC message received from MSC event 120

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	120	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, or SHUTDDIR	8

The following table identifies the parameter list contents associated with the MSC message sent to remote IMS Connect event.

Table 443. MSC message sent to remote IMS Connect event 121

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	121	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	26	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data.	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, SHUTDDIR, or ERRORRSP	8
VAR_LCLPLKID	MSC LCLPLKID name	8
VAR_LINK	LINK name	8

The following table identifies the parameter list contents associated with the MSC message received from remote IMS Connect event.

Table 444. MSC message received from remote IMS Connect event 122

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	122	2
	Reserved	2

Table 444. MSC message received from remote IMS Connect event 122 (continued)

Parameter list item	Content	Length in bytes
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	26	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, SHUTDDIR, or ERRORRSP	8
VAR_LCLPLKID	MSC LCLPLKID name	8
VAR_PARTNER	MSC partner ID name	8

The following table identifies the parameter list contents associated with the MSC message sent to MSC event.

Table 445. MSC message sent to MSC event 123

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	123	2
	Reserved	2
EVENT_KEY	Session token or EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	10	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_MSGTYPE	REQUEST, REQRESP, RESTART, RSTRESP, RSTBWRSP, PST/SBI, PST/BIS, SHUTDDIR, ERRORSP, or ICONTERM	8

The following table identifies the parameter list contents associated with the Connection to remote IMS Connect timed out event.

Table 446. Connection to remote IMS Connect timed out event 124

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	124	2
	Reserved	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2

Table 446. Connection to remote IMS Connect timed out event 124 (continued)

Parameter list item	Content	Length in bytes
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with the start of session event.

Table 447. Start of session event 125

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	125	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	10	2
VAR_DATA	Start of variable data	0
VAR_APAR	APAR sequence number	2
VAR_TOKEN	SVT token value	8

The following table identifies the parameter list contents associated with the end of session trigger event.

Table 448. End of session trigger event 126

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	126	2
	Reserved	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	0	2

The following table identifies the parameter list contents associated with the establishment of a socket connection with a remote CICS subsystem.

Table 449. Socket connected on RMTCICS event 256

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	256	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2

Table 449. Socket connected on RMTCICS event 256 (continued)

Parameter list item	Content	Length in bytes
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with the disconnection of a socket from a remote CICS subsystem.

Table 450. Socket disconnected on RMTCICS event 257

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	257	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with IMS Connect refreshing a RACF user ID.

Table 451. IMS Connect refreshed a RACF user ID event 258

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	258	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	14	2
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_UIDRFRSHD	Refreshed RACF user ID	8
VAR_RACF_RSN	Reason from the RACF security server for refreshing the ID. This value is passed to IMS Connect in the IRR_ENF2Q field of the RACF parameter list for ENF event 71.	4

The following table identifies the parameter list contents associated with IMS Connect sending a health status report to Work Load Manager (WLM).

Table 452. IMS Connect sent a health status report to WLM event 259

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	259	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	0	2
VAR_DATA_LL	6	2
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_HLTHVAL	Health status sent	4

The following table identifies the parameter list content associated with the PORT resource created event.

Table 453. PORT resource created event 771

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	771	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the PORT resource deleted event.

Table 454. PORT resource deleted event 772

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	772	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the PORT resource updated event.

Table 455. PORT resource updated event 773

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	773	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the DATASTORE resource created event.

Table 456. DATASTORE resource created event 774

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	774	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the DATASTORE resource deleted event.

Table 457. DATASTORE resource deleted event 775

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	775	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the DATASTORE resource updated event.

Table 458. DATASTORE resource updated event 776

Parameter list item	Content	Length in bytes
TOKEN	Token address	4

Table 458. DATASTORE resource updated event 776 (continued)

Parameter list item	Content	Length in bytes
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	776	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the IMSPLEX resource created event.

Table 459. IMSPLEX resource created event 777

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	777	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list content associated with the IMSPLEX resource deleted event.

Table 460. IMSPLEX resource deleted event 778

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	778	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of RSIB	4

The following table identifies the parameter list contents associated with the start of a communication thread for a connection with a remote CICS subsystem.

Table 461. Communication thread started for an RMTCICS connection event 2050

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2

Table 461. Communication thread started for an RMTCICS connection event 2050 (continued)

Parameter list item	Content	Length in bytes
EXTD_EVENT_NUMBER	2050	2
EVENT_KEY	EVNT	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	0	2
EVENT_DATA_ADDR	Address of the TCPIB	4

The following table identifies the parameter list contents associated with receiving an ISC message from IMS.

Table 462. ISC message received from IMS event 2051

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	2051	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	20	2
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_ISFLTYPE	IS field type	2
VAR_MSGTYPE	Message type	8
VAR_ASTOKEN	Associated event token if available when MSGTYPE=CAPEXREQ, CAPEXRSP, BISREQ, or BISRSP	8

The following table identifies the parameter list contents associated with the sending of an ISC message to IMS.

Table 463. ISC message sent to IMS event 2052

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	2052	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	20	2

Table 463. ISC message sent to IMS event 2052 (continued)

Parameter list item	Content	Length in bytes
EVENT_DATA_ADDR	Address of the DSIB	4
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_ISFLTYPE	IS field type	2
VAR_MSGTYPE	Message type	8
VAR_ASTOKEN	Associated event token if available when MSGTYPE=CAPEXREQ, CAPEXRSP, BISREQ, or BISRSP	8

The following table identifies the parameter list contents associated with receiving an ISC message from CICS on an RMTCICS socket connection.

Table 464. ISC message received on RMTCICS socket connection event 2053

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	2053	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	12	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_ISFLTYPE	IS field type	2
VAR_MSGTYPE	Message type	8

The following table identifies the parameter list contents associated with sending an ISC message to CICS on an RMTCICS socket connection.

Table 465. ISC message sent on RMTCICS socket connection event 2054

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	2054	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	28	2

Table 465. ISC message sent on RMTCICS socket connection event 2054 (continued)

Parameter list item	Content	Length in bytes
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_ISFLTYPE	IS field type	2
VAR_MSGTYPE	Message type	8
VAR_ISCNODE	ISC node	8
VAR_ISCUSER	ISC user	8

The following table identifies the parameter list contents associated with receiving an ISC message from CICS on a CICSPORT socket connection.

Table 466. ISC message received on CICSPORT socket connection event 2055

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	2055	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	12	2
EVENT_DATA_ADDR	Address of the TCPIB	4
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_ISFLTYPE	IS field type	2
VAR_MSGTYPE	Message type	8

The following table identifies the parameter list contents associated with sending an ISC message to CICS on a CICSPORT socket connection.

Table 467. ISC message sent on CICSPORT socket connection event 2056

Parameter list item	Content	Length in bytes
TOKEN	Token address	4
EVENT_NUMBER	255	2
EXTD_EVENT_NUMBER	2056	2
EVENT_KEY	Session token	8
DATA_ADDR_COUNT	1	2
VAR_DATA_LL	28	2
EVENT_DATA_ADDR	Address of the TCPIB	4

Table 467. ISC message sent on CICSPORT socket connection event 2056 (continued)

Parameter list item	Content	Length in bytes
VAR_DATA	Start of variable data area	0
VAR_APAR	APAR sequence number for the control block	2
VAR_ISFLTYPE	IS field type	2
VAR_MSGTYPE	Message type	8
VAR_ISCNODE	ISC node	8
VAR_ISCUSER	ISC user	8

Control blocks and DSECTS for event recording

Each table in this section lists the DSECTS and describes the parameter list contents of control blocks that are used to record an IMS Connect event and associated data.

Event recording parameter list (ERPL)

The ERPL is used to record an IMS Connect event and associated data to an event-recording log.

The parameter list contains mandatory and optional fields. The content and usage of the list arguments is dependent on the event being recorded. The DSECT name is HWSERPL. The contents of the ERPL which are pointed to by HWSTECL0 are shown in the following table.

Table 468. Event recording parameter list (ERPL) pointed to by HWSTECL0

Element	Length	Usage and meaning
TOKEN	4	Address of the token for event recording. This is the token returned in the EICB when event recording was initialized. Required.
EVENT_NUMBER	2	A number that identifies the type of a recorded event. Required.
EXTENDED_EVENT_ NUMBER	2	An additional event number that is included in events that have an event number of 255. For events that include an extended event number, the extended event number identifies the type of event.
EVENT_KEY	8	The event key that is associated with the event being recorded. Required.
DATA_ADDR_COUNT	2	Count of the number of EVENT_DATA_ADDR entries in the parameter list. A count of 0 indicates that no entries are present. Required, but can be 0.
VAR_DATA_LL	2	Length of the variable data element. The variable data length does not include this length field. A length of 0 indicates that no variable data is present. Required, but can be 0.
EVENT_DATA_ADDR	4	The address of a data element that begins with a two-byte length field. The parameter list can contain any number of element addresses. The number of element addresses is contained in DATA_ADDR_COUNT. Optional.

Table 468. Event recording parameter list (ERPL) pointed to by HWSTECL0 (continued)

Element	Length	Usage and meaning
VAR_DATA	VAR	A variable length field containing event dependent data. The length of the data element is defined by VAR_DATA_LL. Only one variable data element can be present in the parameter list. Optional.

Event Interface Control Block (EICB)

The EICB links IMS Connect and the trace and event recording module, HWSTECL0.

The block is formatted by IMS Connect and passed to HWSTECL0 with the initialization request. The DSECT name is HWSEICB.

The contents of the EICB are shown in the following table.

Table 469. EICB parameter list contents

Element	Length	Usage and meaning
EYECATCHER	4	Value of EICB identifying this block in working storage. Set by caller.
FLAGS	1	Interface control flags: 1. Event recording is enabled.
EVENT_TOKEN	4	Address of the token used by the event recording routine. The token must be passed to the event recording routine when an event-recording request is made.
EVENT_ADDRESS	4	Entry address of event recording routine.
	4	Reserved space.
	4	Reserved space.
MESSAGE_LEN	2	Length of the message returned from HWSTELC0 module.
MESSAGE_AREA	120	An area that can be used by HWSTECL0 to return an informational or error message to IMS Connect.

TCP/IP information block (TCPIB)

The TCPIB information block is used to pass information about TCP/IP events to the event recording routine.

The block contains a length field that allows the recording routine to capture the block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record based on the length field. The DSECT is HWSTCPIB.

The contents of the TCPIB are shown in the following table.

Table 470. TCP/IP information block (TCPIB) contents

Element	Offset (Hex.)	Length	Usage and meaning
LENGTH	0	2	Length of the TCPIP block, including the length of the field.
BLOCK_ID	2	1	Block ID = X'01' identifying the block as a TCPIB.
	3	1	Reserved.

Table 470. TCP/IP information block (TCPIB) contents (continued)

Element	Offset (Hex.)	Length	Usage and meaning
VERSION	4	2	The version and release of IMS Connect in the VVRR format.
APAR_COUNT	6	2	A sequential count field starting at one and incrementing by one for any APAR changing the format or content of the control block. The number resets to one at each new release. For IMS Version 11, the APAR count is 2. For IMS Version 12, the APAR count is 1.
PORT_NUMBER	8	2	The port number associated with the TCP/IP event being recorded.
SOCKET_NUM	A	2	The socket number associated with the request. This field is redefined to LOCAL_PC_NUM. A 2-byte reserved field follows this field to account for the 4-byte length of LOCAL_PC_NUM.
	C	2	Unmapped bytes.
LOCAL_PC_NUM	A	4	For the local option only, the PC number used by the local connection.
SOCKET_FLAG	E	1	A flag byte identifying information about the socket: **X'80'** Listen socket **X'40'** Session socket
PORT_FLAG	F	1	A flag byte identifying information about the port. **X'80'** SSL port **X'40'** Local port **X'20'** DRDA port **X'10'** Send port **X'08'** CICS port
LENGTH_ISSUED	10	4	The length values associated with the read or write command.
LENGTH_EXECUTED	14	4	The length value actually executed by the read or write command.
LOCAL_SND_LEN	10	4	Overlays LENGTH_ISSUED. For the local interface, the lengths for a local send operation.
LOCAL_RCV_LEN	14	4	Overlays LENGTH_EXECUTED. For the local interface, the lengths for a local receive operation.

Table 470. TCP/IP information block (TCPIB) contents (continued)

Element	Offset (Hex.)	Length	Usage and meaning
EVENT_DATA	18	4	Data or flag bits or both associated with the event. This data can be unique for each event recording the TCPIB.
RETURN_CODE	1C	4	Return code associated with the request.
TCPIP REASON_CODE	20	4	Reason code received from TCP/IP.
	24	4	Unmapped bytes.
LOCAL REASON_CODE	20	8	Reason code received from local interface.
The name of this field varies depending on the value of the PTFLAG field: • SENDCLNT • RECVCLNT	28	8	The contents of this field differ depending on the value of the PTFLAG field. If the value of the PTFLAG field is X'10' (data store), then this field is named SENDCLNT and contains the ID of the send client socket. For all other PTFLAG values, if this field is present, it is named RECVCLNT and contains the ID of the receive client socket.
RMTCONN	30	8	The name of the remote connection as defined on the RMTCICS statement.

Data store information block (DSIB)

The DSIB is used to pass information about data store-related events to the event recording routine.

The DSIB is also used with the SYSPLEX interface. The block contains a length field that allows the recording routine to capture the block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record. The DSECT name is HWSDSIB.

The contents of the DSIB are shown in the following table.

Table 471. Data store information block (DSIB) contents

Element	Offset (Dec.)	Length	Usage and meaning
LENGTH	0	2	Length of the DSIB block, including the length of the field.
BLOCK_ID	2	1	Block ID = X'02' identifying the block as a DSIB.

Table 471. Data store information block (DSIB) contents (continued)

Element	Offset (Dec.)	Length	Usage and meaning
DS_FLAG	3	1	A flag byte providing information about the DSTOR_NAME field. The value of this field determines the type of conditional parameters that are stored in the fields starting at offset 40. See the descriptions of conditional parameters that are listed towards the end of this table. The DS_FLAG element can have the following values: **X'80'** Name is data store **X'40'** Name is SCI **X'20'** Name is MEMBER **X'10'** Name is TMEMBER **X'08'** Name is ODBM **X'04'** Name is MSC **X'02'** Name is ISC
VERSION	4	2	The version and release of IMS Connect in the VVRR format.
APAR_COUNT	6	2	A sequential count field starting at one and incrementing by one for any APAR changing the format or content of the control block. The number resets to one at each new release.
DSTOR_NAME	8	16	Name associated with the data store. For the SYSPLEX (SCI) interface, this is the SYSPLEX name. The field can also be the name of a MEMBER or TMEMBER.
DATA_LEN	24	4	Length associated with a send or receive operation.
DATA_ADDR	28	4	Data address if any associated with the event. Currently, only OTMA sends and receives operations.
RETURN_CD	32	4	The return code associated with the operation.
REASON_CD	36	4	The reason code associated with the operation.
COND_PARMS	40	64	Conditional parameters. The contents of this field depend on the values of the DS_FLAG and CFLAG1 fields.
CFLAG1	104	1	Common FLAG1. The value of this field determines the type of conditional parameters that are stored in the fields starting at offset 40. See the descriptions of conditional parameters that are listed towards the end of this table. The CFLAG1 element can have the following values: **X'80'** The data store was dynamically added.
CFLAG2	105	1	Common FLAG2.

Table 471. Data store information block (DSIB) contents (continued)

Element	Offset (Dec.)	Length	Usage and meaning
	106	2	Reserved.
DSID	108	8	ID that identifies the data store resource that can be defined by the ID parameter of the DATASTORE statement in the HWSCFGxx member of the IMS PROCLIB data set, or by the NAME() parameter in the **CREATE IMSCON TYPE(DATASTORE)** command.
	116	12	Reserved.
Conditional parameters when DS_FLAG is X'08' (ODBM)			
DSIB_DLIFUNC	40	4	DLI function code if any
DSIB_PSBNAME	44	8	PSBname if available
DSIB_APSBTKN	52	16	APSB token
Conditional parameters when DS_FLAG is X'80' (Data store)			
TPIPE_NAME	40	8	TPIPE name that is associated with a data transfer.
CUR_SVTTOKEN	48	8	SVT Token associated with the request if a token value exists.
Conditional parameters when DS_FLAG is X'04' (MSC)			
LINK	40	8	MSC link name that is associated with a data transfer.
Conditional parameters when DS_FLAG is X'02' (ISC)			
NODENAME	40	8	ISC link name that is associated with a data transfer.
ISCUSER	48	8	ISC user name that is associated with a data transfer.
Conditional parameters when CFLAG1 is X'80' (Data store dynamically added)			
MEMBER	40	16	IMS Connect XCF member name.
SMEMBER	56	4	OTMA super member name.
APPL	60	8	RACF application name.
GROUP	68	8	XCF group name.

Resource information block (RSIB)

The Resource information block (RSIB) is used to pass information about resource change events to the event recording routine. It contains a resource information section and a chain of attribute entries that were changed.

The block contains a length field that allows the recording routine to capture the block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record. The DSECT name is HWSRSIB.

The contents of the RSIB are shown in the following table.

Table 472. Resource information block (RSIB) contents

Element	Offset (Dec.)	Length	Usage and meaning
LENGTH	0	2	Length of the RSIB block, including this length field and the chain of attribute entries.
BLOCK_ID	2	1	Block ID = X'03', identifying the block as an RSIB.
	3	1	Reserved.
VERSION	4	2	Version and release of IMS Connect in the VVRR format.
APAR_COUNT	6	2	Sequential count field, starting at one and incrementing by one for any APAR changing the format or content of the control block.
RESOURCE_NAME	8	8	Resource name.
RESOURCE_TYPE	16	16	Resource type.
ACTION_TYPE	32	8	Action type that was performed on the resource: **CREATE** Resource was created. **DELETE** Resource was deleted. **UPDATE** Resource was updated.
	40	12	Reserved.
ATTRIBUTE_ENTRIES_LEN	52	2	Total length of attribute entries. The value is zero if there are no entries.
ATTRIBUTE_ENTRIES_OFFSET	54	2	Offset to chain of attribute entries. The value is zero if there are no entries.
COMMAND_TIMESTAMP	56	8	Timestamp of the command. This timestamp is the same value as the event key used for events 67 (message that is sent to SCI) and 68 (message that is received from SCI) for type-2 commands.
	64	64	Reserved.

The ATTRIBUTE_ENTRIES_OFFSET field in the RSIB points to the chain of attribute entries. The RSIB address, plus the ATTRIBUTE_ENTRIES_OFFSET value, is the address of the first attribute entry. Each attribute entry is mapped by the DSECT ATR. The total length of all the attribute entries is in the ATTRIBUTE_ENTRIES_LEN field.

The existence of an attribute entry indicates that the attribute is changed. If no attributes are changed, then there are no attribute entries, and the ATTRIBUTE_ENTRIES_LEN and ATTRIBUTE_ENTRIES_OFFSET fields are zero.

The contents of an attribute entry (ATR) are shown in the following table.

Table 473. RSIB attribute entry (ATR) contents

Element	Offset (Dec.)	Length	Usage and meaning
LENGTH	0	2	Length of the attribute entry, including this length field and the attribute value.
FLAG1	2	1	Flag1 byte: **X'80'** Character value. **X'40'** Decimal value.
	3	1	Reserved.
ATTRIBUTE_ID	4	8	Attribute identifier.
ATTRIBUTE_VALUE	12	variable	Attribute value. The length of the attribute value is variable. The type of value is defined by the character or decimal bits in the FLAG1 byte field. The length of the attribute value is the value of LENGTH for this entry, minus the 12-byte header.

The following is an example of the RSIB passed to the 771 event that is triggered by the command:
CREATE IMSCON TYPE(PORT) NAME(8001) SET(KEEPAV(5000) EDITRTN(HWSPIOX0))

```
00000 17C10BC0   00B80300 14100001 F8F0F0F1 40404040    | ........8001     |
00010 17C10BD0   D7D6D9E3 40404040 40404040 40404040    | PORT             |
00020 17C10BE0   C3D9C5C1 E3C54040 00000000 00000000    | CREATE  ........ |
00030 17C10BF0   00000000 00380080 2CF26316 DC0C18C0    | .........2.....{ |
00040 17C10C00   00000000 00000000 00000000 00000000    | ................ |
00050 17C10C10   00000000 00000000 00000000 00000000    | ................ |
00060 17C10C20   00000000 00000000 00000000 00000000    | ................ |
00070 17C10C30   00000000 00000000 00000000 00000000    | ................ |
00080 17C10C40   00148000 D7D6D9E3 E3E8D7C5 D9C5C740    | ....PORTTYPEREG  |
00090 17C10C50   40404040 00104000 D2C5C5D7 C1E54040    |     .. .KEEPAV   |
000A0 17C10C60   00001388 00148000 C5C4C9E3 D9E3D540    | ...h....EDITRTN  |
000B0 17C10C70   C8E6E2D7 C9D6E7F0 00000000 00000000    | HWSPIOX0........ |
```

Security Information Block (SAFIB)

The SAFIB is used to pass information about security-related events to the event-recording routine.

The block contains a length field that allows the recording routine to capture block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record. The DSECT name is HWSSAFIB.

The contents of SAFIB are shown in the following table.

Table 474. Security Information Block (SAFIB) contents

Element	Length	Usage and meaning
LENGTH	2	Length of the SAFIB block, including the length of the field.
BLOCK_ID	1	Block ID = X'03' identifying the block as a SAFIB.
VERSION	2	The version and release of IMS Connect in the VVRR format.
APAR_COUNT	2	A sequential count field starting at one and incrementing by one for any APAR changing the format or content of the control block. The number resets to one at each new release.

Table 474. Security Information Block (SAFIB) contents (continued)

Element	Length	Usage and meaning
REQUEST_TYPE	1	Flag indicating the type of request: 1. Type is VERIFY. 2. Type is FASTAUTH. 3. Type is DELETE. 4. Type is LIST. 5. Type is R_PASSWORD.
USERID	8	USERID or PASSTICKET associated with the request.
CLASS_NAME	8	Name of the SAF class associated with the request.
RETURN_CODE	4	Return code received.
REASON_CODE	4	Reason code received from the SAF interface.

Variable Data Block (VDB)

The VDB is used to present variable data to the event-recording interface.

The block is contained within the event parameter list. The block does not contain a length field. The length of this block is specified in the even parameter lists. This allows the block information to be captured regardless of the content or length. When the block is recorded, the entire block is moved to the event record.

The DSECT name for events 1 to 245 is HWSVDB*xx*, where *xx* is the event number. The DSECT name for events 255 and higher is HWSV*xxxx*, where *xxxx* is the four-digit event number.

The contents of the VDB are shown in the following table.

Table 475. Variable Data Block (VDB) contents

Element	Length	Usage and meaning
VAR_DATA	variable	A set of fields defined as variable data for each event that contains variable data. Each event can have individually defined variable data.

DSECTs for event recording

Macros are shipped with IMS Connect to help customize with event recording.

The following table lists all the macros:

Table 476. Event recording macros shipped with IMS Connect

Macro	Function
HWSDSIB	DATA STORE INFORMATION BLOCK
HWSEICB	EVENT INITIALIZATION BLOCK
HWSERPL	EVENT RECORDING PARAMETER LIST
HWSRSIB	RESOURCE INFORMATION BLOCK
HWSSAFIB	SAF INTERFACE BLOCK
HWSTCPIB	TCPIP EVENT INFORMATION BLOCK
HWSV0258	EVENT 258 VARIABLE DATA BLOCK

Table 476. Event recording macros shipped with IMS Connect (continued)

Macro	Function
HWSV0259	EVENT 259 VARIABLE DATA BLOCK
HWSV2051	EVENT 2051 VARIABLE DATA BLOCK
HWSV2052	EVENT 2052 VARIABLE DATA BLOCK
HWSV2053	EVENT 2053 VARIABLE DATA BLOCK
HWSV2054	EVENT 2054 VARIABLE DATA BLOCK
HWSV2055	EVENT 2055 VARIABLE DATA BLOCK
HWSV2056	EVENT 2056 VARIABLE DATA BLOCK
HWSVDB01	EVENT 01 VARIABLE DATA BLOCK
HWSVDB02	EVENT 02 VARIABLE DATA BLOCK
HWSVDB03	EVENT 03 VARIABLE DATA BLOCK
HWSVDB04	EVENT 04 VARIABLE DATA BLOCK
HWSVDB06	EVENT 06 VARIABLE DATA BLOCK
HWSVDB08	EVENT 08 VARIABLE DATA BLOCK
HWSVDB11	EVENT 11 VARIABLE DATA BLOCK
HWSVDB13	EVENT 13 VARIABLE DATA BLOCK
HWSVDB21	EVENT 21 VARIABLE DATA BLOCK
HWSVDB23	EVENT 23 VARIABLE DATA BLOCK
HWSVDB26	EVENT 26 VARIABLE DATA BLOCK
HWSVDB27	EVENT 27 VARIABLE DATA BLOCK
HWSVDB29	EVENT 29 VARIABLE DATA BLOCK
HWSVDB33	EVENT 33 VARIABLE DATA BLOCK
HWSVDB35	EVENT 35 VARIABLE DATA BLOCK
HWSVDB37	EVENT 37 VARIABLE DATA BLOCK
HWSVDB38	EVENT 38 VARIABLE DATA BLOCK
HWSVDB40	EVENT 40 VARIABLE DATA BLOCK
HWSVDB42	EVENT 42 VARIABLE DATA BLOCK
HWSVDB44	EVENT 44 VARIABLE DATA BLOCK
HWSVDB46	EVENT 46 VARIABLE DATA BLOCK
HWSVDB47	EVENT 47 VARIABLE DATA BLOCK
HWSVDB48	EVENT 48 VARIABLE DATA BLOCK
HWSVDB49	EVENT 49 VARIABLE DATA BLOCK
HWSVDB50	EVENT 50 VARIABLE DATA BLOCK
HWSVDB51	EVENT 51 VARIABLE DATA BLOCK
HWSVDB61	EVENT 61 VARIABLE DATA BLOCK

Table 476. Event recording macros shipped with IMS Connect (continued)

Macro	Function
HWSVDB62	EVENT 62 VARIABLE DATA BLOCK
HWSVDB69	EVENT 69 VARIABLE DATA BLOCK
HWSVDB70	EVENT 70 VARIABLE DATA BLOCK
HWSVDB71	EVENT 71 VARIABLE DATA BLOCK
HWSVDB72	EVENT 72 VARIABLE DATA BLOCK
HWSVDB81	EVENT 81 VARIABLE DATA BLOCK
HWSVDB82	EVENT 82 VARIABLE DATA BLOCK
HWSVDB83	EVENT 83 VARIABLE DATA BLOCK
HWSVDB84	EVENT 84 VARIABLE DATA BLOCK
HWSVDB85	EVENT 85 VARIABLE DATA BLOCK
HWSVDB86	EVENT 86 VARIABLE DATA BLOCK
HWSVDB87	EVENT 87 VARIABLE DATA BLOCK
HWSVDB89	EVENT 89 VARIABLE DATA BLOCK
HWSVDB90	EVENT 90 VARIABLE DATA BLOCK
HWSVDB91	EVENT 91 VARIABLE DATA BLOCK
HWSVDB92	EVENT 92 VARIABLE DATA BLOCK
HWSVDB93	EVENT 93 VARIABLE DATA BLOCK
HWSVDB94	EVENT 94 VARIABLE DATA BLOCK
HWSVDB95	EVENT 95 VARIABLE DATA BLOCK
HWSVDB96	EVENT 96 VARIABLE DATA BLOCK
HWSVDB97	EVENT 97 VARIABLE DATA BLOCK
HWSVDB98	EVENT 98 VARIABLE DATA BLOCK
HWSVDBA0	EVENT 100 VARIABLE DATA BLOCK
HWSVDBA1	EVENT 101 VARIABLE DATA BLOCK
HWSVDBA2	EVENT 102 VARIABLE DATA BLOCK
HWSVDBA3	EVENT 103 VARIABLE DATA BLOCK
HWSVDBA4	EVENT 104 VARIABLE DATA BLOCK
HWSVDBA7	EVENT 107 VARIABLE DATA BLOCK
HWSVDBA8	EVENT 108 VARIABLE DATA BLOCK
HWSVDBB0	EVENT 110 VARIABLE DATA BLOCK
HWSVDBB1	EVENT 111 VARIABLE DATA BLOCK
HWSVDBB2	EVENT 112 VARIABLE DATA BLOCK
HWSVDBB4	EVENT 114 VARIABLE DATA BLOCK
HWSVDBB5	EVENT 115 VARIABLE DATA BLOCK

Table 476. Event recording macros shipped with IMS Connect (continued)

Macro	Function
HWSVDBB7	EVENT 117 VARIABLE DATA BLOCK
HWSVDBB8	EVENT 118 VARIABLE DATA BLOCK
HWSVDBC1	EVENT 121 VARIABLE DATA BLOCK
HWSVDBC2	EVENT 122 VARIABLE DATA BLOCK
HWSVDBC4	EVENT 124 VARIABLE DATA BLOCK

Terminating HWSTECL0

To end event recording, IMS Connect calls the event recording routine address in the EICB.

The routine is passed to the ERPL, which defines the event and event data. The event number which is passed to the event recording routine corresponds to the Connect Region Termination event.

When the termination processing for event recording has completed, HWSTECL0 must return to the caller otherwise IMS will hang.

Note: The termination call to HWSTECL0 is made even if the event recording flag in the EICB is not on. If the EICB contains a token and event recording address, the termination call is made so that event recording can terminate the event recording environment.

The event recording termination call can only occur when the caller is executing under the JOBSTEP TCB, the caller is in primary TCB mode, and all tasks as potential event records have terminated.

Related reference

"Event record formats" on page 700
The IMS Connect Event Recorder exit routine stores and categorizes the event and format for all event records using the ERPL (Event Record Parameter List).

IMS Connect Password Change exit routine (HWSPWCH0)

The IMS Connect Password Change exit routine (HWSPWCH0) is an object-code-only (OCO) module that processes password change requests that are passed to it from the HWSSMPL0, HWSSMPL1, or HWSJAVA0 exit routines.

This topic contains Product-sensitive Programming Interface information.

The HWSPWCH0 exit routine validates the format of the password change request before issuing a RACF call to change the password. If an error is detected, HWSPWCH0 sets the error code, message text, message length, SAF return code, RACF return code, and RACF reason code in appropriate fields defined in HWSIMSEA.

The object code for HWSPWCH0 resides in ADFSLOAD member of the Distribution (DLIB) data set.

To enable the HWSPWCH0 exit routine, include the HWSPWCH0 object code and specify the statement INCLUDE TEXT(HWSPWCH0) in the bind JCL of either HWSSMPL0, HWSSMPL1, or HWSJAVA0.

The following JCL binds the object code to enable client password change.

```
//HWSSMPL JOB (ACTINF01),'PGMRNAME',
//            CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),RECION=4M
//SMPL01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=SYS1.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=IMSHWS.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSSMPL0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
```

```
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMSBLD.IMSCON22.APAR.MAINT.SHWSSRC(HWSSMPL0),DISP=SHR
//* Put your HWSSMPL0 source code here
//SMPL02 EXEC PGM=IEWL,
//        PARM='SIZE=(180K,28K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSBLD.USERTEMP.HWSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
//* Put HWSPWCH0 object code here
 INCLUDE   TEXT(HWSSMPL0)
 INCLUDE   TEXT(HWSPWCH0)
 ENTRY   HWSSMPL0
 MODE RMODE(24),AMODE(31)
 NAME HWSSMPL0(R)
//
```

Part 6. REXX SPOC exit routine

The REXX SPOC exit routine provides processing options to manage multiple REXX applications that run in parallel under the same address space.

Chapter 17. CSLULXD0 user exit

Use the CSLULXD0 exit to enable serialization for the IMS REXX SPOC subcommand environment for multiple REXX applications that are running in parallel.

About this routine

You might use the CSLULXD0 exit and the SEROPT keyword to set the system level specifications if you need to serialize multiple parallel REXX programs that start the IMS REXX SPOC command environment in the same address space.

You are not required to use the CSLULXD0 exit. If you do not use the exit or if the exit cannot be loaded at initialization time, and the keywords are not specified on the LINK CSLULXSB REXX statement, default values are used and the serialization is not enabled.

If you use the CSLULXD0 exit, ensure that you assemble and link the CSLULXD0 exit into a data set specified in the STEPLIB concatenation. A sample CSLULXD0 exit is provided in the IMS.ADFSSMPL data set.

The following JCL sample assembles and binds the CSLULXD0 exit into the IMS.SDFSRESL data set:

```
//ASSEMBLE EXEC PGM=ASMA90,PARM='NOOBJ,DECK'
//SYSLIB DD DSN=IMS.SDFSMAC,DISP=SHR
//SYSPUNCH DD DISP=OLD,DSN=IMS.OBJDSET(CSLULXD0)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),SPACE=(CYL,(15,15))
//SYSIN    DD *
ULXD0    TITLE 'CSLULXD0 - IMS Rexx SPOC DEFAULTS BLOCK'
CSLULXD0 CSECT
         SPACE 1
         CSLULXD TYPE=BEGIN
         CSLULXD TYPE=PARM,SEROPT=ENQ
*****    CSLULXD TYPE=PARM,SEROPT=NONE
         CSLULXD TYPE=END
         END   CSLULXD0
//STEP1 EXEC PGM=IEWL,
//             PARM='SIZE=(880K,64K),NCAL,LET,REUS,XREF,LIST'
//SYSPRINT DD  SYSOUT=*
//SYSPUNCH DD DSN=IMS.OBJDSET,DISP=SHR
//SYSLMOD  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSPUNCH)),SPACE=(CYL,(10,1))
//SYSLIN   DD *
 INCLUDE SYSPUNCH(CSLULXD0)
 NAME CSLULXD0(R)
```

Keyword of the exit

Use the **CSLULXD** macro to specify IMS REXX SPOC options in the CSLULXD0 exit. You can specify the following keyword:

SEROPT=NONE|ENQ

Specifies whether a serialization enqueue (ENQ) needs to be obtained to serialize parallel instances of the IMS REXX SPOC command environment that are started under the same address space.

You can use the **SEROPT** keyword in the CSLULXD0 exit to set the system level specifications. The default value is **SEROPT=**NONE. If this keyword is specified, the specified value becomes the system specification. If this keyword is not specified or the CSLULXD0 exit is not available, the default value is used for the system level specification.

Important: Each REXX SPOC instance that is in the same address space must use the same serialization option. When serialization is required,**SEROPT=**ENQ (either in the LINK CSLULXSB command, or in the CSLULXD0 exit) must be specified by every REXX SPOC instance in the same address space.

You can specify the following values for this keyword:

ENQ

SEROPT=ENQ indicates that multiple IMS REXX SPOC instances might be started under the same address space and a serialization ENQ is required to serialize the subcommand environment.

The ENQ option specifies the STEP option so that the scope is limited to the address space. The ENQ resource qname is *CSLUSPCA*, and the rname is *CSLULXTP||jobid*. You can use the **D GRS,RES=(CSLUSPCA,*)** command to display serialization information.

When you set up an IMS REXX subcommand environment, a step level ENQ for resource CSLULXTP||jobid is obtained. This ENQ will not be released until the IMS REXX **END** subcommand is processed.

Important: Ensure that either the REXX SPOC application always issues an IMS REXX **END** subcommand to dequeue instances, or that the started task ends in a timely manner, if the IMS REXX subcommand environment is started to prevent subsequent instances in the same address space from waiting on the ENQ.

NONE

This is the default value. **SEROPT=**NONE indicates that multiple IMS REXX SPOC instances are not started under the same address space and a ENQ serialization is not required.

Enabling and disabling serialization for parallel REXX applications

You can enable the serialization option for REXX SPOC applications by specifying the **SEROPT** parameter in two ways:

- At system level, specify the CSLULXD macro in the CSLULXD0 exit. For example, specify CSLULXD TYPE=PARM,SEROPT=ENQ to enable serialization.
- At application level, specify the **SEROPT** parameter on the "LINK CSLULXSB" statement. This specification could overwrite the default or system level specification. For example, specify LINK CSLULXSB SEROPT=ENQ to enable serialization.

The following examples show how to enable or disable the serialization by specifying the **SEROPT** parameter:

Enabling serialization for parallel REXX applications with SEROPT=ENQ

```
/*-------------------------------------------------------------------
| When multiple IMS Rexx SPOC instances could be started           |
| under the same address space, then an ENQ is needed to           |
| serialize the subcommand environment.                            |
| The Rexx SPOC program requests this by specifying                |
| SEROPT=ENQ on the Address LINK 'CSLULXSB' statement.             |
-------------------------------------------------------------------*/
Address LINK 'CSLULXSB SEROPT=ENQ'
if rc = 0 then
  do
    Address IMSSPOC

    "IMS  IPLX4"
    "ROUTE IMS1,IMSB"

    "WAIT  5:00"

    "CART DISTRAN"
    "/DIS TRAN PART"

     spoc_rc = cslulgts('DISINFO.','DISTRAN',"59")
     do z1 = 1 to  DISINFO.0
       /* display each line of XML information */
       Say disinfo.z1
     end
     "END"
  End
```

Disabling serialization for parallel REXX applications with SEROPT=NONE

```
/*-------------------------------------------------------------------
| When only a single IMS Rexx SPOC instance is started             |
| under an address space, then an ENQ is not needed to             |
| serialize the subcommand environment.                            |
| The Rexx SPOC program requests this by specifying                |
| SEROPT=NONE on the Address LINK 'CSLULXSB' statement.            |
| This is the default (SEROPT can be omitted).                     |
-------------------------------------------------------------------*/
Address LINK 'CSLULXSB SEROPT=NONE'
if rc = 0 then
  do
    Address IMSSPOC

    "IMS  IPLX4"
    "ROUTE IMS1,IMSB"

    "WAIT  5:00"

    "CART DISTRAN"
    "/DIS TRAN PART"

     spoc_rc = cslulgts('DISINFO.','DISTRAN',"59")
     do z1 = 1 to  DISINFO.0
       /* display each line of XML information */
       Say disinfo.z1
     end
     "END"
  End
```

Related concepts

Setting up the REXX environment in a CSL (System Programming APIs)

Part 7. TSO SPOC user exit routines

TSO SPOC user exit routines run in ISPF and pass information through the ISPF environment.

The exit routines can use the ISPF commands VGET and VPUT to view and alter variables used by the TSO SPOC.

Chapter 18. EXITPGM user exit

Before TSO SPOC starts, you must specify the EXITPGM user exit. The EXITPGM command can send a single exit or a list of user exits.

Program exits are called with a z/OS batch program parameter list. All command exits are called before program exits.

On entry to the routine, register 1 points to the parameter list, which is a standard parameter list. Register 1 points to a fullword, which points to the half-word length followed by the parameter string. The contents of the registry are controlled by ISPF.

The following example shows an example of using the EXITPGM exit routine:

```
DFSSPOC EXITPGM(UEP1, UEP2)
```

The values specified in the parameter list are the names of the input user exits. Each exit can view or change variables in the ISPF shared pool.

Contents of registers

Register
Contents

1
Points to the address of the parameter data (from the PAR keyword) field (halfword length) followed by the data

2-12
Not used

13
72-byte save area

14
Return address

15
Entry address / Return code on exit

Note: The names of the exit routines are standard 1- to 8-character module names. REXX program names can also be specified. A REXX program name can be prefixed by a percent sign (%).

Chapter 19. EXITCMD user exit

Before TSO SPOC starts, you must specify the EXITCMD user exit. The command can send a single exit or a list of user exits.

Command exits are called before user exits.

On entry to the routine, register 1 points to the parameter list. The parameters list is a Command Processor Parameter List (CPPL). Register 1 is defined by macro IKJCPPL. The contents of the registry are controlled by ISPF.

Contents of registers

Register
Contents

1
Points to a CPPL, which is a list of four addresses that point to the following parameters:

- Command buffer
- UPT
- PSCB
- ECT

2-12
Not used

13
72-byte save area

14
Not applicable

15
Return code on exit

Chapter 20. Variables in the ISPF shared pool

The TSO SPOC exit routines can use the ISPF commands **VGET** and **VPUT** to view and alter variables in the ISPF shared pool.

The following table provides details about the variables in the ISPF shared pool that can be changed by using the TSO SPOC exit routines:

Table 477. Variables in the ISPF shared pool

Variable name	Usage	Pool	Description
EXITTYPE	input	shared	A read-only variable that indicates the function type for this call to the exit. 1 indicates a command pre-submit exit.
PLEX	input	shared	A 1- to 5-character name of the IMSplex.
ROUTE	input	shared	A 1- to 8-character name of the IMSplex member that the command will be sent. A list of names separated by commas can also be used. The list can be as long as 1024 characters.
CMDTEXT	input and output	shared	The command string that is routed to OM in the IMSplex. The maximum length is 32760.
RTC	output	shared	Return code set by the user exit. • RTC = 0 - No changes to the command parameters • RTC = 4 - User Exit modified one or more command parameters • RTC = 8 - Reject the command parameters **Note:** The input user exit must set the RTC = 4 if any of the command parameters are changed and must set the message text to describe what was changed.
RSN	output	shared	Reason code set by the user exit.
MSGTEXT	output	shared	A text message passed by the user exit to notify the SPOC user of any changes, including the message id, when the RTC = 4 or 8. The maximum length is 256 characters.

Related concepts

z/OS: Using variables

Chapter 21. REXX program example using the EXITCMD exit routine

The following example shows how to retrieve and update variables in the ISPF shared pool by using the EXITCMD exit routine.

In the following REXX program, the EXITCMD exit routine checks the routing information and rejects a command sent to IMS1 (production IMS).

```
"VGET (EXITTYPE) SHARED" If exittype = 1 then
    Do
        "VGET (ROUTE) SHARED"
        If pos("IMS1", route)> 0 Then RTC = 8
        MSGTEXT = "REJECTED - IMS1 IS RESTRICTED FOR PRODUCTION USE"
        "VPUT (RTC, MSGTEXT) SHARED
    End
```

The original command parameters and changes made by the user exit with a return code 4 or 8 are logged in an ISPF log file.

After the REXX program is executed, the following example shows an ISPF log file in which the **QUERY TRAN** command was rejected:

```
NAME        EXITTYPE        PLEX    ROUTE    CMDTXT        RTC    RSN    MSGTXT
-------- ---------- ----- ----- ---------- ---- ---- -------------
ORIGINAL  1         PLEX1 IMS1  QRY TRAN   0     0
USEREXIT1 1         PLEX1 IMS2  QRY TRAN   4     0   CHANGED ROUTE
USEREXIT2 1         PLEX2 IMS2  QRY TRAN   8     0   REJECTED
```

Related concepts

z/OS: Using variables

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

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, General-use Programming Interface and Associated Guidance Information, and 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.

General-use programming interfaces allow the customer to write programs that obtain the services of IMS. General-use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a section or topic or by a General-use programming interface label.

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 15.2 library.

Title	Acronym
IMS Version 15.2 Application Programming	APG
IMS Version 15.2 Application Programming APIs	APR
IMS Version 15.2 Commands, Volume 1: IMS Commands A-M	CR1
IMS Version 15.2 Commands, Volume 2: IMS Commands N-V	CR2
IMS Version 15.2 Commands, Volume 3: IMS Component and z/OS Commands	CR3
IMS Version 15.2 Communications and Connections	CCG
IMS Version 15.2 Database Administration	DAG
IMS Version 15.2 Database Utilities	DUR
IMS Version 15.2 Diagnosis	DGR
IMS Version 15.2 Exit Routines	ERR
IMS Version 15.2 Installation	INS
IMS Version 15.2 Licensed Program Specifications	LPS
IMS Version 15.2 Messages and Codes, Volume 1: DFS Messages	MC1
IMS Version 15.2 Messages and Codes, Volume 2: Non-DFS Messages	MC2
IMS Version 15.2 Messages and Codes, Volume 3: IMS Abend Codes	MC3
IMS Version 15.2 Messages and Codes, Volume 4: IMS Component Codes	MC4
IMS Version 15.2 Operations and Automation	OAG
IMS Version 15.2 Release Planning	RPG
IMS Version 15.2 System Administration	SAG
IMS Version 15.2 System Definition	SDG
IMS Version 15.2 System Programming APIs	SPR
IMS Version 15.2 System Utilities	SUR

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Product Number: 5635-A06
5655-DS5
5655-TM4