

Query Management Facility  
Version 12 Release 1

*Installing and Managing QMF for TSO  
and CICS*



**Note**

Before using this information and the product it supports, be sure to read the general information under "Notices" at the end of this information.

**March 30, 2020 edition**

This edition applies to Version 12 Release 1 of IBM<sup>®</sup> Db2 Query Management Facility (QMF) Classic Edition and Enterprise Edition, which are features of IBM Db2 12 for z/OS (5650-DB2) and IBM Db2 11 for z/OS (5615-DB2). It also applies to Version 12 Release 1 of IBM Db2 QMF for z/OS (5697-QM2), which is a stand-alone IBM Db2 for z/OS tool. This information applies to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 1982, 2020.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© **Rocket Software, Inc. 2013, 2020.**

---

# Contents

<b>About this information.....</b>	<b>xi</b>
What you should know before you begin.....	xi
Service updates and support information.....	xii
Highlighting conventions.....	xii
How to read syntax diagrams.....	xii
How to send your comments.....	xiii
<b>Part 1. Installing QMF for TSO and CICS.....</b>	<b>1</b>
Chapter 1. Planning your configuration.....	3
Supported configurations.....	3
Databases that support QMF Version 12.1.....	3
Installation modes.....	4
Coexistence of releases.....	4
QMF in distributed data networks.....	5
Remote unit of work.....	5
Distributed unit of work.....	8
Chapter 2. QMF installation overview and roadmaps.....	11
QMF installation process overview.....	11
General installation concepts.....	12
Required authorities for installing and administering QMF.....	12
Required authorities for installing QMF.....	13
Required authorities for QMF administration.....	13
Roadmap 1: Installing QMF in stand-alone or requester databases (Db2 for z/OS only).....	14
Installation path A: Installing QMF V12.1 in a Db2 for z/OS stand-alone or requester database.....	16
Installation path B: Migrating to QMF V12.1 from QMF V11.2, V11.1, V10 or a QMF NFM release in a Db2 for z/OS stand-alone or requester database.....	18
Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command.....	19
Installation path C: Installing QMF V12.1 in a Db2 for z/OS, iSeries, or LUW server.....	22
Installation path D: Migrating to QMF V12.1 from QMF V11.2, QMF V11.1, QMF V10, or a QMF NFM release in a Db2 for z/OS, iSeries, or LUW server.....	23
Installation path E: Migrating to QMF V12.1 from V7.2 or earlier in a DB2 for VSE and VM server.....	25
Installation path F: Migrating to QMF V12.1 from QMF V8.1, V9.1, V10.1, V11.1, or V11.2 in a DB2 for VSE and VM server.....	26
Roadmap 3: Installing server databases accessed by three-part names.....	27
Installation path G: Preparing a remote server to be accessed by QMF commands that include three-part names.....	28
Roadmap 4: Databases accessed in languages other than English.....	29
Tasks to perform when upgrading Db2 for z/OS but not QMF.....	30
Chapter 3. Installing or migrating QMF in requester (Db2 for z/OS) databases.....	33
Preparing to install .....	33
Implementing installation prerequisites.....	33
Copying the QMF Version 12.1 libraries from the distribution media.....	34
Populating the VSAM panel library.....	37
Addressing storage requirements.....	37
Moving modules to enhance performance.....	38

Setting installation job parameters for requester installations.....	40
Running the installation jobs.....	45
Jobs that install QMF V12.1 where no prior release is installed.....	45
Jobs that migrate QMF V8 NFM, QMF V9 NFM, QMF V10, QMF V11.1, or QMF V11.2 to QMF V12.1.....	47
Defining programs, resources, and preferences to TSO and CICS.....	49
Customizing requester installations under TSO.....	49
Customizing requester installations under CICS.....	55
Chapter 4. Installing or migrating QMF in server databases.....	61
Setting installation job parameters for server installations.....	61
Running installation jobs that prepare servers for access by the QMF CONNECT command.....	62
Jobs that install QMF V12.1 in server databases where no prior release is installed.....	62
Jobs that migrate QMF V8 NFM, QMF V9 NFM, QMF V10, QMF V11.1, or QMF V11.2 to QMF V12.1 in a z/OS, iSeries, or LUW server.....	64
Jobs that migrate QMF V7.2 or earlier to QMF V12.1 in a VM or VSE server.....	66
Jobs that migrate QMF V8, V9, V10, V11.1 or V11.2 to QMF V12.1 in a VM or VSE server.....	66
Falling back to the prior release in server databases.....	67
Running installation jobs that prepare servers to be accessed by QMF commands that include three-part names.....	68
Chapter 5. Verifying that QMF is installed correctly.....	71
Testing QMF installations on requester databases.....	71
Starting QMF.....	71
Checking for the correct version of the QMF panel library.....	75
Checking that QMF libraries have been allocated correctly.....	77
Checking that all required QMF views were installed.....	77
Installing sample queries and procedures required for the installation verification procedures.....	77
Running the installation verification procedures.....	80
Testing QMF installations on server databases.....	83
Chapter 6. Migration and fallback considerations.....	85
Accessing profiles and objects from a prior QMF release.....	85
Accessing QMF profiles in a different database.....	85
Migrating tables, views, and QMF objects.....	86
Object compatibility between releases.....	86
Forward compatibility of earlier releases with QMF Version 12 Release 1.....	86
Compatibility of QMF Version 12.1 with earlier releases.....	87
Above the bar storage requirement changes.....	88
Chapter 7. Deleting an earlier release of QMF.....	89
Deleting QMF from a Db2 for z/OS database.....	89
When the old and new release are in the same subsystem.....	89
When the old and new release are in different subsystems.....	92
Deleting QMF from a DB2 for iSeries or LUW database.....	93
When the old and new releases are in the same server.....	94
When the old and new release are in different servers.....	94
Deleting QMF from a DB2 for VSE and VM database.....	95
When the old and new releases are in the same server.....	95
When the old and new release are in different servers.....	96
Chapter 8. Installing extra value features.....	97
Configuring QMF Analytics for TSO.....	97
Load library considerations for QMF Analytics for TSO.....	97
Allocating files used by QMF Analytics for TSO.....	97
File system customization for QMF Analytics for TSO.....	98
Installing QMF Analytics for TSO sample tables.....	99

Verifying that QMF Analytics for TSO installed correctly .....	99
Configuring QMF Data Service (QDS).....	100
Installing QMF National Language Features.....	101
Migrating QMF National Language Features.....	106
Installing the QMF stored procedure interface (TSO only).....	107
Installing the enhanced LIST command function (z/OS only).....	109
Configuring the QMF Data Service Interface.....	112

## **Part 2. Managing QMF for TSO and CICS..... 115**

Chapter 9. Starting QMF.....	117
Starting QMF on TSO.....	117
Allocating necessary files and resources.....	117
Adding QMF CLIST and EXEC libraries to TSO.....	117
Starting QMF with the TSO CALL command.....	117
Starting QMF directly with the DSQQMFE module.....	118
Starting QMF in a batch environment.....	119
Starting QMF as a Db2 for z/OS stored procedure.....	119
Starting QMF on ISPF.....	124
Starting QMF from an ISPF menu.....	125
Using LIBDEF statements to allocate QMF program libraries.....	126
Starting QMF in batch mode in ISPF.....	128
Examples of starting QMF under ISPF.....	128
Starting QMF in native z/OS batch.....	129
Starting QMF on CICS .....	130
Initializing global variables and QMF session behavior when QMF starts.....	131
Setting global variables with the DSQUOPTS routine.....	132
Setting global variables with the global variable table.....	134
Initialization with the default system initialization procedure.....	137
Creating your own initialization procedure.....	138
Chapter 10. Setting program parameters and preferences at startup time.....	141
Summary of program parameters.....	141
Setting database and environment parameters.....	148
Specifying the name of the Db2 for z/OS subsystem in which to start QMF under TSO.....	148
Specifying the initial database connection.....	149
Specifying the name of the QMF application plan.....	149
Specifying which ID to use as the QMF profile key under TSO.....	150
Defining storage for reports.....	150
Defining a fixed amount of virtual storage for reports.....	151
Defining a variable amount of virtual storage for reports.....	151
Acquiring extra storage for data no longer needed in virtual storage.....	152
Controlling performance of fetch and insert operations.....	159
Controlling report wait time.....	159
Enabling support for multirow fetch and insert.....	160
Using multiple database threads.....	161
Automating QMF activity.....	162
Specifying the mode of operation (interactive or batch).....	162
Specifying an initial procedure to run when QMF starts.....	163
Setting tracing options.....	167
Setting the trace for TSO.....	168
Setting the trace for CICS.....	168
Printing DBCS data from non-DBCS display devices.....	169
Chapter 11. Registering users and setting privileges.....	171
Controlling access to the application plan and packages.....	171
Granting access to the application plan and packages.....	171

Revoking access to the application plan and packages.....	171
Creating QMF user profiles.....	172
About the Q.PROFILES table.....	172
Establishing a profile structure for your site.....	180
Adding a user profile.....	181
Updating a user profile.....	183
Deleting a user profile.....	184
Providing access to QMF and database objects.....	185
Privileges required for QMF commands and functions.....	185
Granting and revoking privileges.....	187
Setting standards for objects and allowing uncommitted read.....	189
Sharing QMF objects with other users.....	189
Allowing uncommitted read.....	189
Users' object lists.....	190
Customizing users' object lists.....	190
Default behavior of the QMF LIST and DESCRIBE commands.....	192
Object list storage requirements.....	195
Chapter 12. Creating and maintaining objects in the database.....	197
Enabling users to create tables in the database.....	197
Procedure for creating tables.....	197
Assigning a table space for SAVE DATA, IMPORT, and RUN QUERY commands.....	198
Granting a user the privileges to create tables.....	199
Using views to filter sensitive data.....	200
Creating a view.....	200
Granting privileges for a view.....	200
Maintaining the QMF object catalog.....	201
Structure of the Q.OBJECT_DIRECTORY table.....	201
Structure of the Q.OBJECT_DATA table.....	204
Structure of the Q.OBJECT_REMARKS table.....	205
Enlarging the table space for the QMF object catalog.....	205
Listing QMF objects.....	207
Displaying QMF objects.....	208
Transferring ownership of QMF objects.....	208
Deleting obsolete QMF objects.....	209
Importing queries, forms, and procedures from z/OS data sets.....	209
Maintaining a Db2 for z/OS subsystem.....	209
Managing data sets.....	210
Maintaining the QMF control tables.....	210
Chapter 13. Setting up printing and charting functions.....	213
Deciding whether to use QMF or GDDM services for printing.....	213
Using GDDM services to handle printing.....	214
How QMF interfaces with your GDDM nickname.....	214
Where GDDM searches for the nickname.....	214
Example nicknames for different printer families.....	214
Example nickname definitions for specific printers.....	215
Setting up GDDM services to handle printing.....	216
Using QMF services to handle printing.....	220
QMF services for printing in native z/OS batch, TSO, and ISPF.....	220
Using QMF services for printing in CICS.....	221
Allowing users to print without exiting QMF.....	222
Printing requirements by object type.....	223
Enabling charting functions.....	224
Enabling chart support in TSO and ISPF.....	225
Enabling chart support in CICS.....	225
Chapter 14. Command synonyms.....	227

Using the default synonyms provided with QMF.....	227
List of default synonyms.....	227
The DPRE synonym: using ISPF to preview the printed report.....	228
Guidelines for synonyms.....	229
Synonym verbs.....	229
Synonym object names.....	230
Synonym definitions.....	231
Customizing Command synonyms.....	234
Creating a command synonym table.....	234
Entering command synonym definitions into the table.....	235
Activating the synonyms.....	236
Minimizing maintenance of command synonym tables.....	237
Assigning one synonyms table to all users.....	237
Assigning views of a synonyms table to individual users.....	237
Chapter 15. Custom function keys.....	239
Customizing QMF function keys.....	239
Displaying the panel ID.....	239
Choosing the keys that you want to customize.....	242
Creating the function key table.....	245
Entering your function key definitions into the table.....	246
Activating new function key definitions.....	248
Testing and diagnosing problems with the function key table.....	249
Examples of key definitions.....	249
Chapter 16. Custom edit exit routines for QMF forms.....	253
Edit exit routines and QMF.....	253
Fields passed to and from the exit routine.....	254
Fields of the interface control block.....	254
Fields that characterize the input and output areas.....	257
Choosing an edit code.....	258
Double-byte character set data and edit routines.....	259
DBCS data and what the edit routine receives.....	259
Ensuring that the edit routine returns the right results.....	260
Date, time, and timestamp data and edit routines.....	260
Required formats for date, time, and timestamp information.....	261
Db2 exits for date and time data.....	262
Edit routines for programming languages.....	263
Writing an edit routine in High-Level Assembler.....	263
Writing an edit routine in PL/I.....	268
Writing an edit routine in COBOL.....	273
Chapter 17. Controlling QMF resources.....	281
The default governor exit routine provided with QMF for TSO and CICS.....	281
How a governor exit routine controls resources.....	281
Resource limits with the default governor exit.....	284
Program components of the governor exit routine.....	289
How QMF and the governor interact.....	290
How and when QMF calls the governor exit routine.....	291
Modifying the default governor exit routine or writing your own routine.....	298
Passing resource control information to the governor exit.....	298
Storing resource control information for the duration of a QMF session.....	310
Providing messages for canceled activities.....	310
Translating, assembling, and link-editing your governor exit routine.....	311
Canceling user activity.....	313
Using the Db2 resource limit facility.....	314
Differences between governors.....	315
QMF commands that can be monitored by the Db2 governor.....	315

How QMF responds when queries are cancelled by the Db2 governor.....	316
Configuring Db2 governing of QMF commands.....	317
Chapter 18. Running QMF in batch mode.....	319
Running QMF as a batch program on z/OS.....	319
Authority to operate in batch mode.....	319
JCL to execute a QMF batch job.....	320
Running QMF batch in native z/OS.....	321
Running QMF batch under TSO.....	323
Running QMF batch under ISPF using the QMF BATCH command.....	324
Initiating a QMF batch job in the foreground on ISPF or TSO.....	331
Debugging a batch-mode procedure or application.....	331
Starting a QMF batch job from a remote Db2 client.....	332
Running QMF as a batch transaction on CICS.....	333
Running batch activities from a terminal.....	333
Running batch activities without a terminal.....	333
Debugging a procedure.....	334
Chapter 19. Troubleshooting and diagnosing problems.....	335
Applying QMF service.....	335
Correcting common problems.....	336
Errors that can occur at initialization time.....	336
Warning messages after you start QMF.....	344
Incorrect output.....	344
Problems with printing.....	345
Display errors.....	347
Resolving storage-related issues.....	347
Managing QMF performance.....	347
Capturing EXPLAIN information for dynamic statements.....	347
Enabling QMF queries to be eligible for query acceleration.....	348
Solving storage problems.....	349
Solving resource contention problems.....	351
Improving QMF performance with the DSQEC_BUFFER_SIZE global variable setting.....	351
Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement.....	352
Using diagnosis aids.....	353
Diagnosing your problem using message support.....	353
Creating an interrupt to capture diagnostic information.....	354
The trace facility.....	356
Examining error log reports.....	362
Using diagnostics native to the environment.....	363
Reporting a problem to IBM.....	365
Searching previously reported problems.....	365
Working with IBM Software Support.....	366
<b>Appendix A. Summary of changes in prior releases.....</b>	<b>367</b>
Version 11.2 changes.....	367
Version 11.1 changes.....	369
Version 10.1 changes.....	374
Version 9.1 changes.....	377
Version 8.1 changes.....	378
Version 7.2 changes.....	379
Version 7.1 changes.....	380
Version 6.1 changes.....	381
<b>Appendix B. QMF functions that require specific support.....</b>	<b>383</b>
Functions that vary according to database type.....	383



Functions not available in CICS.....	384
<b>Appendix C. QMF objects that reside in Db2.....</b>	<b>387</b>
QMF plans.....	387
QMF packages.....	387
QMF control tables and table spaces for TSO and CICS.....	387
QMF views.....	388
Db2 for z/OS storage groups.....	389
Space for saving LOB data.....	390
VSAM clusters for TSO/CICS.....	390
QMF sample tables.....	391
<b>Appendix D. External editors.....</b>	<b>393</b>
Modifying QMF queries and procedures with an editor.....	393
Inserting QMF reports into documents.....	394
Changing the application.....	395
Renaming the document interface macro (DSQAED1P).....	395
Placing the Q.DSQAED1S procedure in the database.....	395
Changing the data components.....	396
Changing the CLISTs and macros.....	397
<b>Appendix E. QMF user-defined functions.....</b>	<b>401</b>
QMF user-defined functions and procedures.....	401
APPL_AUTHNAMES.....	401
CALL DSQABA1E.....	402
DSQABA1E.....	403
<b>Appendix F. How QMF and GDDM programs are defined to CICS.....</b>	<b>405</b>
How QMF programs are defined to CICS.....	405
How GDDM definitions are loaded during installation of QMF.....	406
Transaction routing to control resource use in CICS.....	406
<b>Notices.....</b>	<b>407</b>
Programming interface information.....	408
Trademarks.....	408
<b>Glossary.....</b>	<b>409</b>
<b>Index.....</b>	<b>423</b>



## About this information

---

IBM Db2® Query Management Facility for TSO and CICS® is a tightly integrated, powerful, and reliable tool that offers query and reporting functions that help you access and present data from any of the following relational databases:

- Db2 for z/OS®
- Db2 for Linux, UNIX, and Windows
- DB2® for iSeries
- DB2 Server for VSE and VM

These topics are designed to help database administrators and systems programmers with:

- Planning for and performing a new QMF installation or migrating to the current QMF release from a previous release
- Starting QMF on a Db2 for z/OS application requester
- Customizing and managing the QMF environment for users
- Creating exit routines that establish resource governing capabilities and support for user-defined data formatting codes
- Running QMF in batch mode
- Troubleshooting and diagnosing problems

## What you should know before you begin

---

You should be familiar with the components that make up your specific environment. These components are listed and explained below.

- The z/OS operating system.
- Time Sharing Option (TSO), an environment that supports QMF and its related products.
- Interactive System Productivity Facility (ISPF), a dialog manager for QMF.
- Customer Information Control System (CICS), a general-purpose data communication and online transaction processing system. CICS provides the interface between QMF and z/OS.
- The base Graphical Data Display Manager (GDDM) product, which makes it possible for QMF to display panels. The base GDDM product can also be used to print reports and other objects. GDDM-PGF is required for creating charts.
- Db2 for z/OS, the database manager for QMF.
- SMP/E (System Modification Program Extended), the tool used to load QMF content from the distribution media to the Db2 for z/OS target system where you will be performing the first QMF installation.
- High-Level Assembler (HLASM), which is needed to modify the default governor exit routine or create your own. HLASM can also be used to create your own edit codes for QMF forms.
- COBOL and PL/I, which can be used to create your own edit codes for QMF forms.
- REXX, which is used to create execs that install QMF.
- WLM, a component of z/OS that provides the ability to run multiple workloads at the same time within one z/OS image or across multiple images.

You should be familiar with the basic functions provided in QMF before you begin the tasks in these topics. To read more about QMF concepts and functions, see the following information:

- , GC27-8876-01

- , SC27-8879-01
- , SC27-8880-01

## Service updates and support information

---

To find service updates and support information, including software fix packs, PTFs, Frequently Asked Questions (FAQs), technical notes, troubleshooting information, and downloads, refer to the following Web page:

[IBM Software Support website](#)

## Highlighting conventions

---

This information uses the following highlighting conventions:

- **Boldface** type indicates commands or user interface controls such as names of fields, folders, icons, or menu choices.
- Monospace type indicates examples of text that you enter exactly as shown.
- *Italic* indicates the titles of other publications or emphasis on significant terms. It is also used to indicate variables that you should replace with a value.

## How to read syntax diagrams

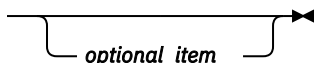
---

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

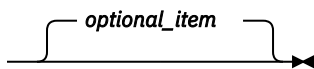
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol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

▶▶ *required\_item* ▶▶

- Optional items appear below the main path.

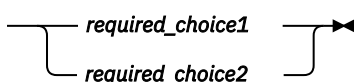
▶▶ *required\_item*  ▶▶

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.

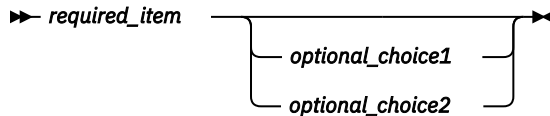
▶▶ *required\_item*  ▶▶

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

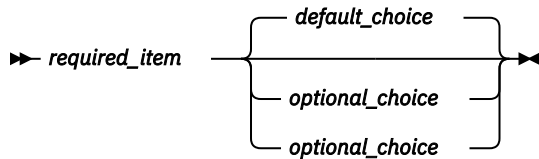
If you *must* choose one of the items, one item of the stack appears on the main path.

▶▶ *required\_item*  ▶▶

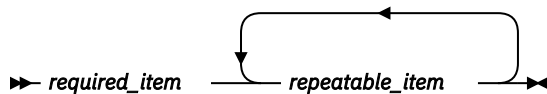
If choosing one of the items is optional, the entire stack appears below the main path.



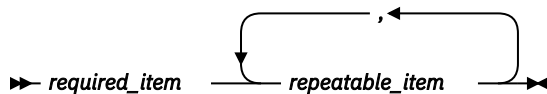
If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



- 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.

- Keywords, and their minimum abbreviations if applicable, appear in upper case. 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).

## How to send your comments

---

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other documentation, use either of the following options:

- Use the online reader comment form, which is located at:

<http://www.ibm.com/software/data/rcf>

- Send your comments by e-mail to [comments@us.ibm.com](mailto:comments@us.ibm.com). Be sure to include the name of the book, the part number of the book, the version of your product, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).



---

# Part 1. Installing QMF for TSO and CICS





# Chapter 1. Planning your configuration

How you install QMF depends on whether a previous release of QMF is installed in the database. Installation also depends on how the target database for QMF will function in your distributed network.

If QMF already exists, you must determine whether migration from the existing release is supported. If QMF does not exist, you must determine whether a new installation is supported for this database type.

You must also consider how the database into which you are installing the product will function in a distributed data configuration. It can stand alone (requiring no connectivity to other databases), function as a requester, function as a server, or function in both capacities.

After you read these topics, move on to the installation roadmaps. Each roadmap asks you a series of questions that leads you to the installation path that is appropriate for your QMF configuration, type of installation or migration, and environment.

## Related concepts

[QMF installation overview and roadmaps](#)

Because you can install QMF in various databases and migrate to QMF Version 12.1 from various versions, roadmaps are provided to guide you through the installation and migration processes.

[Required authorities for installing and administering QMF](#)

Specific authorities are required for installing QMF and for general QMF administration.

## Supported configurations

This topic lists databases that support installation of or migration to QMF Version 12.1. It also provides information about migrating from earlier installation modes and which versions of QMF can coexist with Version 12.1 in the same Db2 subsystem.

### Databases that support QMF Version 12.1

Db2 releases support specific installations of QMF Version 12.1.

The following table shows the supported installations for QMF Version 12.1 according to database type.

For information about compatibility of QMF Version 12.1 with Db2 for z/OS releases not shown in the table, see <http://www.ibm.com/support/docview.wss?uid=swg21409518>. In some cases, QMF PTFs might need to be applied for QMF to operate properly.

Minimum database release	New installations (no prior release of QMF exists)	Migrations from QMF 8.1 NFM, QMF 9.1 NFM, and QMF 10.1 and later
One of the following: <ul style="list-style-type: none"><li>DB2 for z/OS Version 9.1 (VUE and MLC) running in New Function Mode with APAR PM45482 applied</li><li>DB2 for z/OS Version 10.1 (VUE and MLC) running in all modes (except CM8, CM8*, ENFM8, and ENFM8*) with APARs PM50434 and PM72274 applied</li><li>Db2 for z/OS Version 11.1 (VUE and MLC)</li><li>Db2 for z/OS V12.1 (VUE and MLC)</li></ul>	X	X

Table 1. Supported installations for QMF Version 12.1 (continued)

Minimum database release	New installations (no prior release of QMF exists)	Migrations from QMF 8.1 NFM, QMF 9.1 NFM, and QMF 10.1 and later
DB2 for iSeries Version 5.4	X	X
Db2 for Linux, UNIX, and Windows Version 9.5	X	X
DB2 Server for VSE and VM Version 7.3		X

## Installation modes

Before QMF Version 10, two installation modes were supported: Compatibility Mode and New Function Mode.

The two installation modes that QMF supported before Version 10 had these characteristics:

- In QMF Compatibility Mode, the QMF object catalog supported authorization IDs up to eight characters and QMF object names up to 18 characters.
- In QMF New Function Mode, the QMF object catalog supported authorization IDs up to 128 characters and QMF object names up to 128 characters.

The QMF object catalog comprises the Q.OBJECT\_DIRECTORY, Q.OBJECT\_DATA, and Q.OBJECT\_REMARKS control tables. These three tables store information about QMF queries, forms, procedures, analytics, and folder objects.

Starting with QMF Version 10, two different installation modes are no longer supported. The QMF Version 12.1 object catalog supports authorization IDs and object names of the same lengths that were supported by QMF Version 10, QMF Version 9 New Function Mode, and QMF Version 8 New Function Mode. This difference in catalog column lengths determines which releases of QMF can coexist in the same database.

QMF Version 11 Release 2 is the last release that supports migration from a QMF release in Compatibility Mode. QMF Version 12 Release 1 does not support migration from QMF Version 9.1 Compatibility Mode, QMF Version 8.1 Compatibility Mode, QMF Version 7.2, or earlier releases. If you are using one of these versions of QMF, you must migrate to New Function Mode prior to migrating to QMF Version 12 Release 1.

### Related reference

#### Coexistence of releases

QMF Version 12.1 can coexist in the same database only with QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, or QMF Version 10.1, QMF Version 11.1, or QMF Version 11.2.

## Coexistence of releases

QMF Version 12.1 can coexist in the same database only with QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, or QMF Version 10.1, QMF Version 11.1, or QMF Version 11.2.

The rules for coexistence of QMF Version 12.1 is due to a difference in the length of authorization IDs and QMF object names that are supported by the QMF catalog. The following table shows these differences in supported lengths.

Table 2. Lengths supported by the QMF object catalog for authorization IDs and QMF object names in QMF Version 12.1 and prior releases

Release of QMF	Maximum authorization ID length supported by QMF catalog	Maximum object name length supported by QMF catalog
<ul style="list-style-type: none"> <li>• QMF Version 12.1</li> <li>• QMF Version 11.2</li> <li>• QMF Version 11.1</li> <li>• QMF Version 10.1</li> <li>• QMF Version 9.1 New Function Mode</li> <li>• QMF Version 8.1 New Function Mode</li> </ul>	128	128
<ul style="list-style-type: none"> <li>• QMF Version 9.1 Compatibility Mode</li> <li>• QMF Version 8.1 Compatibility Mode</li> <li>• QMF Version 7.2 or earlier</li> </ul>	8	18

## QMF in distributed data networks

The first installation of QMF must be performed in a Db2 for z/OS database. When this installation is complete, the Db2 for z/OS database can then stand alone, function as a requester, or function as both a requester and server for other QMF Version 12.1 installations.

If the QMF installation under Db2 for z/OS will function as a requester, it can be used to access data from any of the following remote database server types:

- Another Db2 for z/OS subsystem
- Db2 for Linux, UNIX, and Windows
- DB2 for iSeries
- DB2 for VSE and VM

QMF supports two methods of remote data access: *remote unit of work* and *distributed unit of work*. Both types of access are based on the definition of a *unit of work*, which is a single logical transaction. A logical transaction consists of a sequence of SQL statements in which either all of the operations are successfully performed or the sequence as a whole is considered unsuccessful. For example, all SQL statements in a multistatement QMF SQL query that contains no COMMIT statements run under a single unit of work.

**Important:** Both types of remote data access require that DRDA communications between the requester and server databases be defined and operational.

**Restriction:** Different databases have varying levels of support for data types and various functions. When you display a table or view on a remote database, some operations such as SAVE or EXPORT might be unsupported if the object includes or refers to a data type that is not supported by that database.

### Remote unit of work

QMF supports the *remote unit of work* of remote data access. *Remote unit of work* is a form of distributed relational database processing in which an application program can access data on a remote database within a unit of work.

A remote unit of work can include more than one relational database request, but all requests must be made to the same remote database. All requests to a database must be completed (either committed or rolled back) before requests can be sent to another database.

When QMF is participating in a remote unit of work configuration, each installation in a Db2 for z/OS database can act as either a requester only or as both a requester and server. Installations of QMF in database types other than Db2 for z/OS can function only as servers, not requesters, in a remote unit of work configuration.

### How the connection is made in remote unit of work configuration

With the remote unit of work data access method, you can connect to the server in one of two ways.

Use one of the following methods to connect from the requester to the server:

- Issue the QMF CONNECT command from within an established QMF session, procedure, or program.
- Specify the DSQSDBNM program parameter when you start QMF. This parameter specifies the name of an initial server to which QMF will connect.

The DSQSDBNM parameter and the DSQSSUBS parameter are used together. QMF first connects to the local database requester that is specified on the DSQSSUBS parameter and then issues a CONNECT command internally to connect to the server location that is specified by the DSQSDBNM parameter, prior to displaying the QMF home panel.

**Restriction:** Connectivity with remote servers is not supported when QMF for TSO is started as a stored procedure in the Db2 for z/OS requester database.

The following figure illustrates QMF in a remote unit of work configuration.

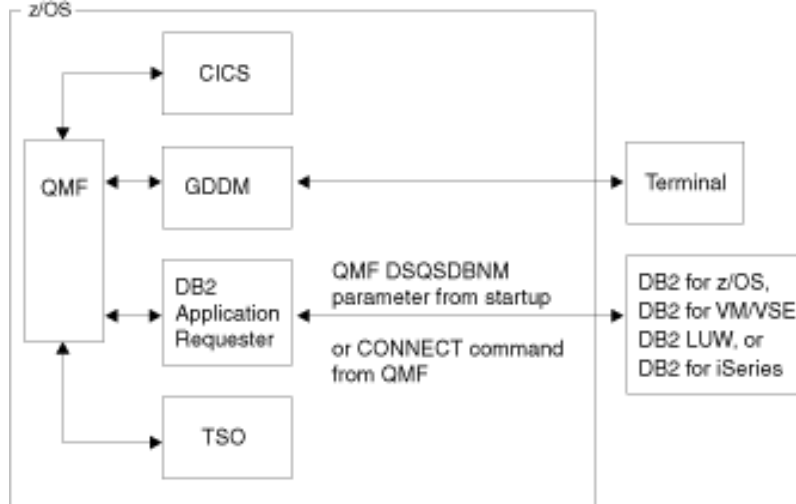


Figure 1. QMF remote unit of work configuration

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### Related reference

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

### What you can access using the QMF CONNECT command

After using the QMF CONNECT command or the DSQSDBNM program parameter to connect to a remote server, you can access both QMF and database objects at the remote server in the same way that you access them locally.

The QMF catalog on DB2 for VSE and VM databases allows authorization IDs to be up to 8 characters and QMF object names to be up to 18 characters. The QMF Version 12.1 catalog on all other database types supports authorization IDs and QMF object names up to 128 characters. When the CONNECT command references a DB2 for VSE and VM database, the differences in the structure of the QMF catalog are

tolerated between the two systems. However, be aware of the supported lengths on the DB2 for VSE and VM system as you run QMF queries and procedures and issue QMF commands.

### **Where QMF needs to be installed for remote unit of work access**

For QMF to connect to a remote server using the QMF CONNECT command or the DSQSDBNM parameter, the same release level of QMF must be present at both the requester and the server.

The following objects must be installed in both locations:

- QMF installation plan and packages
- QMF application packages
- QMF control tables
- QMF catalog views
- Table space for QMF SAVE DATA and IMPORT TABLE commands
- QMF sample tables (optional)

The QMF application plan must be present at the requester only.

If the QMF control tables and packages are on the remote server and are at the same release level as they are on the requester, the connection process is successful and processing continues on the remote server after the connection is established.

### **Related reference**

QMF objects that reside in Db2

These QMF objects are necessary to run QMF Version 12.1 in a Db2 for z/OS subsystem. You can use this information as a guide during recovery operations if necessary.

### **Roadmaps for remote unit of work configurations**

Two procedures are required to set up QMF to connect to a remote server through the QMF CONNECT command.

To set up QMF to use the QMF CONNECT command to access data from one or more remote servers:

1. Follow [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)”](#) on page 14 to install QMF in the requester database.

Only Db2 for z/OS databases can function as requesters. Use roadmap 1 for each requester that must access QMF Version 12 Release 1 on a remote server.

2. Follow [“Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command”](#) on page 19 to install QMF in each remote server database so that each server is enabled for access by QMF CONNECT commands that are issued from requester databases.

If the remote server is a Db2 for z/OS database that is functioning as both a server and a requester, the installation paths in roadmap 1 prepare the database to function in both capacities.

When you finish any of the installation paths that are associated with roadmap 2, the server is also prepared for access by three-part names in QMF commands.

**Restriction:** QMF commands with three-part names cannot be directed to DB2 for VSE and VM servers.

### **Related concepts**

Distributed unit of work

QMF supports the *distributed unit of work* of remote data access. *Distributed unit of work* is a form of distributed relational database processing that enables a user or application program to read or update data at multiple locations within a unit of work.

### **Example configuration for remote unit of work**

This example demonstrates a remote unit of work configuration in which both the server and requester are Db2 for z/OS databases.

In this configuration, the following architecture is used:

- The z/OS operating system z/OS1 has two Db2 12 for z/OS subsystems: DB2A and DB2B. This system is a TSO system; DB2A is an application requester and DB2B is an application server.
- The z/OS operating system z/OS2 has one Db2 12 for z/OS subsystem, DB2C. This system is a batch system; DB2C is an application server, which is accessible to the TSO users on z/OS1.
- QMF must be installed into DB2A as an application requester, and into DB2B and DB2C as an application server. Authorized users on DB2A can access data that is stored at DB2B and DB2C without logging onto different z/OS operating systems.

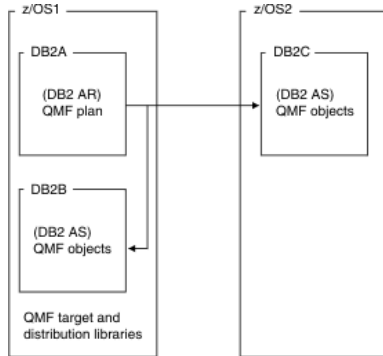


Figure 2. QMF in a remote unit of work configuration where both the requester and server are Db2 for z/OS

## Distributed unit of work

QMF supports the *distributed unit of work* of remote data access. *Distributed unit of work* is a form of distributed relational database processing that enables a user or application program to read or update data at multiple locations within a unit of work.

Within one unit of work, an application running in one system can direct SQL requests to multiple remote database management systems by using the SQL that is supported by those systems. For example, a shop inventory program can update an inventory table on one system and an accounts receivable table on another system within one unit of work.

When QMF is participating in a distributed unit of work configuration, the database that initiates the data request is always Db2 for z/OS, because that is where QMF is running. Database types other than Db2 for z/OS can function only as servers, not requesters.

### Related concepts

#### QMF in distributed data networks

The first installation of QMF must be performed in a Db2 for z/OS database. When this installation is complete, the Db2 for z/OS database can then stand alone, function as a requester, or function as both a requester and server for other QMF Version 12.1 installations.

#### How the connection is made in distributed unit of work configuration

QMF supports distributed unit of work through the use of aliases or three-part names in QMF commands.

Three-part names take the following form:

```
location_name.auth_ID.object_name
```

In this syntax:

- *location\_name* specifies the name of the database that is the target of the request.
- *auth\_ID* specifies the authorization ID of the user who created the object.
- *object\_name* specifies the name of the table or view to be accessed.

#### Restrictions:

- Connectivity with remote servers is not supported when QMF for TSO is started as a stored procedure in the Db2 for z/OS requester database.
- QMF commands that include three-part names cannot be directed to DB2 for VSE and VM servers.

- By default, QMF commands that include three-part names cannot be used to access remote tables that contain large object (LOB) data. To enable access of LOB data in remote tables with three-part names, the DSQEC\_LOB\_RETRV global variable must be set to 3. The DSQEC\_LOB\_RETRV can also be set to 2 to enable the retrieval of LOB metadata only.
- Unless the command that includes the three-part name is directed to Db2 for z/OS, QMF must be started with multirow fetch turned off.

### **Related concepts**

#### Enabling support for multirow fetch and insert

The DSQSMRFI parameter controls whether the database uses multirow or single-row fetch and insert.

### **What you can access using three-part names**

Three-part names in QMF commands allow you to access only database tables and views. To access QMF objects, use a remote unit of work configuration.

To access QMF objects (queries, forms, procedures, analytics, and folder objects) on a remote server, you first must use the QMF CONNECT command to connect to the server database, then issue the request. The QMF CONNECT command requires a remote unit of work configuration.

### **Related concepts**

#### Remote unit of work

QMF supports the *remote unit of work* of remote data access. *Remote unit of work* is a form of distributed relational database processing in which an application program can access data on a remote database within a unit of work.

### **Where QMF needs to be installed for distributed unit of work access**

You can install QMF in two ways to enable users to use aliases or three-part names in QMF commands.

To enable users to use aliases or three-part names in QMF commands, you must complete one of the following tasks:

- Fully install QMF at both the requester and server.
- Fully install QMF at the requester and perform a three-part-name installation at the server.

### **Related tasks**

#### Running installation jobs that prepare servers to be accessed by QMF commands that include three-part names

This job sequence prepares both the requester and server to use three-part names in QMF commands to access data at a remote server.

### **Roadmaps for distributed unit of work configurations**

How you set up QMF for the distributed unit of work method depends on whether you intend to use the QMF CONNECT command with three-part names.

If you intend to use three-part names, but not the QMF CONNECT command for remote data access, complete the following steps:

1. Follow [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) to install QMF in the requester database.
2. Follow [“Roadmap 3: Installing server databases accessed by three-part names” on page 27](#) to prepare the remote server for access by QMF three-part names.

If you intend to use the QMF CONNECT command in addition to three-part names for remote data access, complete the following steps:

1. Follow [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) to install QMF in the requester database.
2. Follow [“Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command” on page 19](#) to install QMF in the remote server. The installation paths that are associated with roadmap 2 prepare the server to be accessed by both the QMF CONNECT command and three-part names.

**Restriction:** QMF commands that include three-part names cannot be directed to DB2 for VSE and VM servers.

**Example configuration for distributed unit of work**

This example configuration shows a Db2 for z/OS-to-Db2 for z/OS distributed unit of work connection.

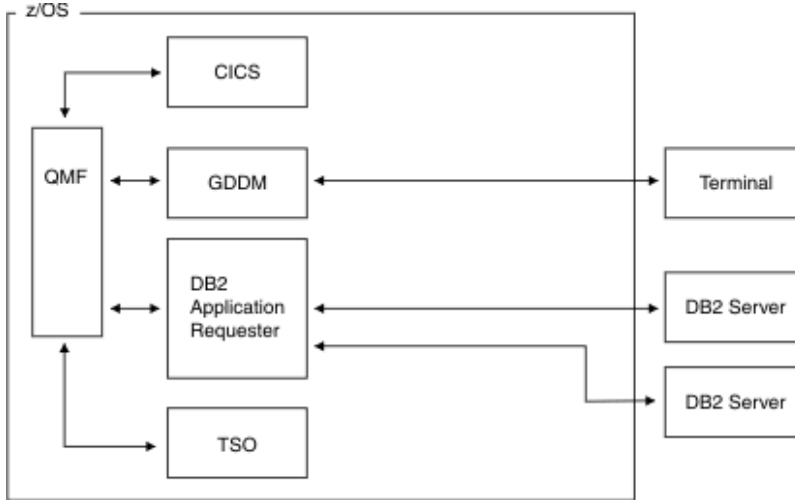


Figure 3. QMF distributed unit of work configuration. (This example shows a Db2 for z/OS-to-Db2 for z/OS configuration.)



## Chapter 2. QMF installation overview and roadmaps

Because you can install QMF in various databases and migrate to QMF Version 12.1 from various versions, roadmaps are provided to guide you through the installation and migration processes.

For example, the steps, requirements, and considerations are different depending on whether you are installing QMF in a stand-alone database, a server database that is accessed by the QMF CONNECT command, or a server database that is accessed by three part names. You can also enable QMF to access data in languages other than English. Roadmaps are provided for each of these options.

**Important:** If you upgrade the database after you install QMF Version 12.1, you must run certain jobs to ensure continued proper operation of QMF.

### QMF installation process overview

The process of installing QMF consists of installing or migrating QMF on databases of your choice and installing any optional features that you want to use.

Step of the installation process	Details for this step
<p><b>1. Install or migrate QMF in the first database.</b></p> <p>This database must be a Db2 for z/OS database.</p> <p>You can install QMF Version 12.1 in any of the following configurations:</p> <ul style="list-style-type: none"><li>• In a stand-alone database.</li></ul> <p>In this configuration, the QMF installation in this database never accesses, or is accessed by, QMF installations in other databases.</p> <ul style="list-style-type: none"><li>• In an application requester in a distributed data configuration.</li></ul> <p>In this configuration, the QMF installation in this database requests data from other QMF installations in remote server databases.</p> <ul style="list-style-type: none"><li>• In both an application requester and an application server in a distributed data configuration.</li></ul> <p>In this configuration, the QMF installation in this database accesses and is accessed by other QMF installations in remote databases.</p>	<p>See roadmap 1 in “<a href="#">Roadmap 1: Installing QMF in stand-alone or requester databases (Db2 for z/OS only)</a>” on page 14.</p>
<p><b>2. Prepare each additional database in your distributed data configuration.</b></p> <p>Where and how you install QMF in a distributed data configuration depends on the configuration of your data network and how you plan to access the data.</p> <ul style="list-style-type: none"><li>• If you plan to use the QMF CONNECT command to access data from remote servers, follow roadmap 2.</li><li>• If you do not plan to use the QMF CONNECT command to connect to this server, but you intend to direct QMF commands with three-part names to this server to access tables and views, follow roadmap 3.</li></ul>	<p>See the following information:</p> <ul style="list-style-type: none"><li>• Roadmap 2 in “<a href="#">Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command</a>” on page 19.</li><li>• Roadmap 3 in “<a href="#">Roadmap 3: Installing server databases accessed by three-part names</a>” on page 27.</li></ul>
<p><b>3. Optional: Configure QMF Analytics for TSO.</b></p> <p>QMF Analytics for TSO provides additional charting and statistical analysis capabilities for users in the TSO environment.</p>	<p>See “<a href="#">Configuring QMF Analytics for TSO</a>” on page 97 for information.</p>

Table 3. An overview of the QMF installation process (continued)	
Step of the installation process	Details for this step
<b>4. Optional: Install QMF National Language Features.</b> A National Language Feature (NLF) allows you to use QMF in a language other than English.	See roadmap 4 in <a href="#">“Roadmap 4: Databases accessed in languages other than English”</a> on page 29.
<b>5. Install other optional features as necessary.</b> <ul style="list-style-type: none"> <li>• QMF for TSO can be started as a Db2 for z/OS stored procedure.</li> <li>• You can also install an enhancement to the LIST command that allows users to list tables and views that are authorized to both primary and secondary authorization IDs. Granting privileges to PUBLIC for these objects is then unnecessary.</li> </ul>	See the following information: <ul style="list-style-type: none"> <li>• <a href="#">“Installing the QMF stored procedure interface (TSO only)”</a> on page 107</li> <li>• <a href="#">“Installing the enhanced LIST command function (z/OS only)”</a> on page 109.</li> </ul>

## General installation concepts

Because QMF for TSO and CICS is a Db2 for z/OS application, you must understand many of the same concepts as you do to install Db2 for z/OS.

Before you begin the installation process, ensure that you have the following knowledge:

- Ensure that you understand the different ways in which QMF can be configured for your environment so that you can make informed decisions. Read [Chapter 1, “Planning your configuration,”](#) on page 3.
- Ensure that you understand the authorization requirements for installing and administering QMF. Read [“Required authorities for installing and administering QMF”](#) on page 12.
- Ensure that you understand the requirements for requester databases. Read [“Installation prerequisites for requester \(Db2 for z/OS\) databases”](#) on page 33.
- If you intend to install QMF on different databases that are part of a distributed data configuration, ensure that you understand the remote unit of work and distributed unit of work concepts. Read [“QMF in distributed data networks”](#) on page 5.
- Ensure that you understand the following concepts:
  - CREATE, INSERT, and GRANT SQL statements, including the meaning of granting privileges or authorities to PUBLIC.
  - The terms *application plan*, *package*, and *bind*.
  - Databases, table spaces, tables, and views.
  - Database security.

### Related concepts

[QMF in distributed data networks](#)

The first installation of QMF must be performed in a Db2 for z/OS database. When this installation is complete, the Db2 for z/OS database can then stand alone, function as a requester, or function as both a requester and server for other QMF Version 12.1 installations.

## Required authorities for installing and administering QMF

Specific authorities are required for installing QMF and for general QMF administration.

### Related concepts

[Installation prerequisites for requester \(Db2 for z/OS\) databases](#)

Before you can install QMF on Db2 for z/OS databases that function as stand-alone or requester databases, you must fulfill the hardware and software requirements.

#### Planning your configuration

How you install QMF depends on whether a previous release of QMF is installed in the database. Installation also depends on how the target database for QMF will function in your distributed network.

## **Required authorities for installing QMF**

Ensure that you have the necessary authorities for the databases in which you will be installing QMF.

The following authorities are required:

- Installations on Db2 for z/OS databases require SYSADM authority.

If your site uses security access control external to Db2 such as RACF, reference installation option EXTSEC in the installation defaults exec QMF1210.SDSQEXCE(DSQ1DEFS). Proper setting of the EXTSEC variable will eliminate SQL GRANT statements from being issued during the installation process. Additionally, the default view job will be created without reference to SYSIBM.SYSTABAUTH.

- Installations on Db2 for Linux, UNIX, or Windows databases require SYSADM authority.
- Installations on iSeries databases require \*ALLOBJ authority.
- Installations on VM or VSE databases require DBA authority.

Regardless of the user ID that is used to install QMF, the following resources are owned by the "Q" authorization ID:

- All QMF control tables
- Sample procedures and queries
- Sample tables
- Default views for the database object list

### **Related concepts**

Installation defaults for common parameters

Default values in the QMF1210.SDSQEXCE(DSQ1DEFS) exec.

### **Related tasks**

Customizing users' object lists

Using the default views supplied by QMF for your table lists and column information can increase processing time, because Db2 gathers authorization information from the SYSIBM.SYSTABAUTH table. If you do not need the extra security provided by these authorization checks, consider creating your own views that generate a list of objects stored in the database.

### **Related reference**

Installing sample queries and procedures required for the installation verification procedures

Install the sample QMF queries and procedures only on TSO.

QMF control tables and table spaces for TSO and CICS

These are the control tables shipped with QMF.

## **Required authorities for QMF administration**

To avoid issuing multiple GRANT statements for specific database objects, you must use an authorization ID with DBADM or equivalent authority for most QMF administration and customization tasks. You need this level of authority on any database in which your users store data.

You need an authorization ID with DBADM or equivalent authority to the following databases on Db2 for z/OS systems:

### **DSQDBCTL**

The QMF control tables are stored in this database.

## DSQDBDEF

When a user issues a SAVE DATA command, the resulting table is stored by default in this database, in table space DSQTSDEF. Depending on your installation, users may be creating tables in implicitly created table spaces in the DSQDBDEF table.

If your users will be working with database and QMF objects in a Db2 for Linux, UNIX, and Windows database, you will need DBADM authority on the DSQTSCTL and DSQTSOBJ database partition groups.

To administer QMF under a different ID from which you installed the product, grant DBADM authority to the new ID.

During initialization, QMF checks the authorization ID of the user who is starting QMF to determine whether that ID has either the INSERT or DELETE privilege on the Q.PROFILES control table. If the authorization ID has either of these privileges, QMF considers the user to be a *QMF administrator*. QMF administrator authority makes it easier for you to administer and maintain QMF because QMF administrators can perform the following commands on QMF queries, forms, and procedures that are owned by other users without requiring the owners to share these objects with all users:

- SAVE
- ERASE
- IMPORT
- EXPORT
- DISPLAY

QMF administrators can also issue the DISPLAY command on ANALYTIC objects and the ERASE command on ANALYTIC and FOLDER objects that are owned by other users.

A user who has DBADM or equivalent authority on the database where the QMF control tables are stored (DSQDBCTL) automatically has QMF administrator authority.

The DSQUOPTS exit routine runs during QMF initialization and provides a way for you to disable QMF administrator authority checking by using the DSQEC\_DISABLEADM global variable. However, if you disable this feature, you must explicitly set up administrator access to users' QMF objects. For example, you can request that users save their objects with the SHARE=YES parameter, or you can use the DSQEC\_SHARE global variable to change the default value for the SHARE parameter to YES. You can also set a new default value for the DSQEC\_SHARE global variable in the DSQUOPTS exit routine.

### Related concepts

[Initializing global variables and QMF session behavior when QMF starts](#)

You can use several methods to set QMF global variables when QMF starts. These methods involve modifying the DSQUOPTS routine, the Q.GLOBAL\_VARS global variable table, or the default system initialization procedure. This procedure can also be used to set other aspects of a user's QMF session before the home panel displays.

### Related reference

[QMF control tables and table spaces for TSO and CICS](#)

These are the control tables shipped with QMF.

## Roadmap 1: Installing QMF in stand-alone or requester databases (Db2 for z/OS only)

---

Use this information for the first QMF installation and for databases that will stand alone, function as requesters only, or function as both requesters and servers. Only Db2 for z/OS databases can stand alone or function as requesters.

### About this task

The roadmap and installation paths for stand-alone or requester databases apply to the following installation types:

- The initial installation of QMF Version 12 Release 1. The first installation must always be in a Db2 for z/OS database because Db2 for z/OS is the only type of database from which QMF can be started.
- QMF installations in databases that will never access remote servers or be accessed by remote requesters. These databases are known as *stand-alone* databases.
- QMF installations in databases that function as requesters only in a distributed data configuration.
- QMF installations in databases that function as both servers and requesters in a distributed data configuration.

### Procedure

Use the following roadmap to identify the installation path that is appropriate for your environment.

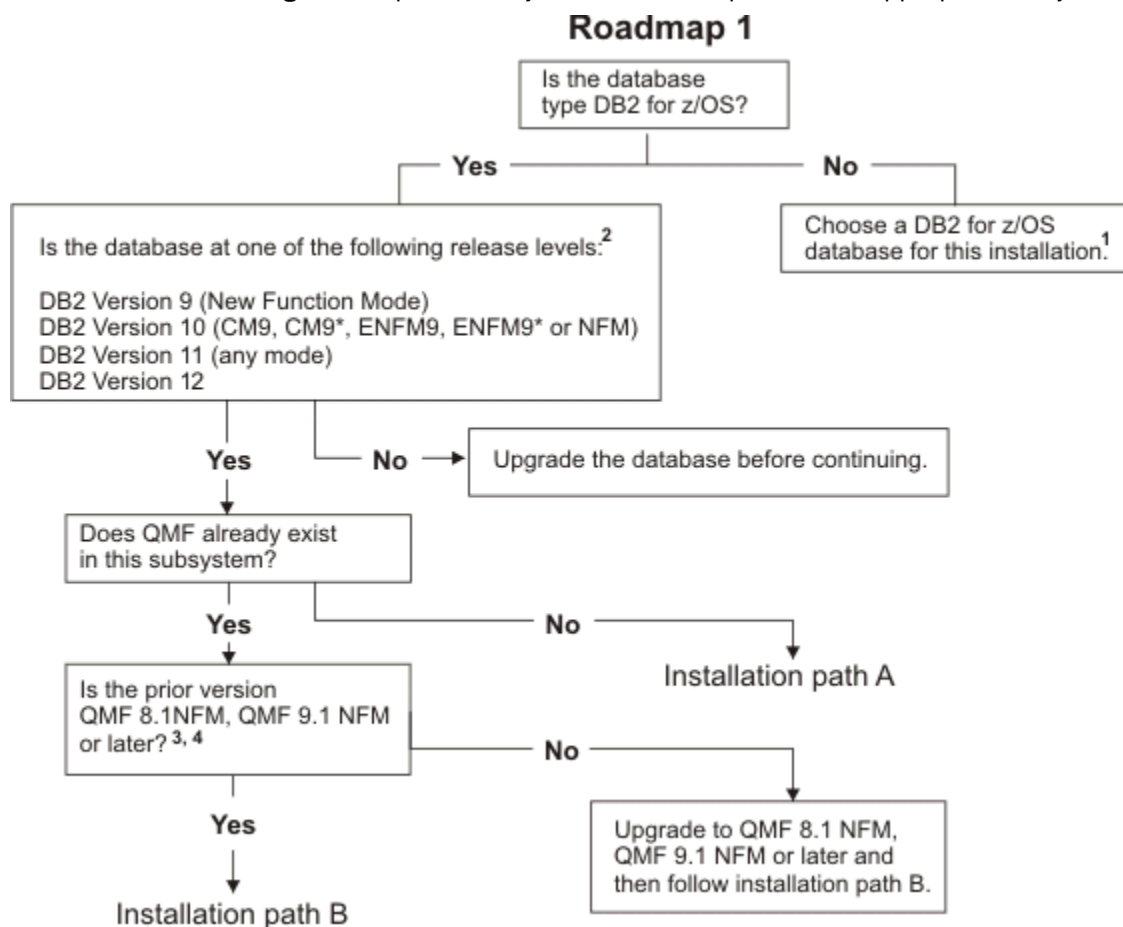


Figure 4. Installation roadmap for the first QMF Version 12 Release 1 installation or databases that will stand alone, function as requesters only, or function as both requesters and servers

### Roadmap notes:

- The first installation must be performed on a Db2 for z/OS database because Db2 for z/OS is the only type of database from which QMF can be started.
- To determine which Db2 for z/OS release is installed, run the following query:

```
SELECT GETVARIABLE('SYSIBM.VERSION')
FROM SYSIBM.SYSDUMMY1
```

The result is DSNvvrrm, where DSN indicates that Db2 for z/OS is the database, vv indicates the version, rr indicates the release level, and m indicates the modification level. For example, a DB2 10 for z/OS database returns the following result:

```
DSN10015
```

To determine if the Db2 installation is Db2 Value Unit Edition, run the SYSPROC.ADMIN\_INFO\_SYSPARM stored procedure and check the value of the OTC\_LICENSE parameter. Db2 VUE installations have a value of TERMS\_ACCEPTED for this parameter; otherwise, the value is NOT\_USED.

- c. To determine which QMF release is installed, start QMF. The version and release are displayed on the QMF home panel.
- d. To determine the mode of an existing QMF installation, run the following query. You can run this query from QMF, SPUFI, or any other SQL tool that is available to you.

```
SELECT LENGTH
FROM SYSIBM.SYSCOLUMNS
WHERE NAME = 'USERID'
AND TBNAME = 'ERROR_LOG'
AND TBCREATOR = 'Q'
```

- If the query returns a value of 8, the existing QMF installation is in Compatibility Mode.
- If the query returns a value of 128, the existing QMF installation is in New Function Mode.
- If the query returns no value, QMF is not installed on the server that you are checking.

### What to do next

Based on your results after you work through this roadmap, proceed to installation path A or B.

### Related concepts

[QMF in distributed data networks](#)

The first installation of QMF must be performed in a Db2 for z/OS database. When this installation is complete, the Db2 for z/OS database can then stand alone, function as a requester, or function as both a requester and server for other QMF Version 12.1 installations.

### Related tasks

[Installation path A: Installing QMF V12.1 in a Db2 for z/OS stand-alone or requester database](#)

Use this procedure for the initial installation of QMF in a Db2 for z/OS stand-alone or requester database. This installation path describes the steps that are required to install QMF Version 12.1 on a Db2 for z/OS database that contains no prior release of QMF.

[Installation path B: Migrating to QMF V12.1 from QMF V11.2, V11.1, V10 or a QMF NFM release in a Db2 for z/OS stand-alone or requester database](#)

Use this procedure to migrate QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, QMF Version 10, or QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1.

## Installation path A: Installing QMF V12.1 in a Db2 for z/OS stand-alone or requester database

Use this procedure for the initial installation of QMF in a Db2 for z/OS stand-alone or requester database. This installation path describes the steps that are required to install QMF Version 12.1 on a Db2 for z/OS database that contains no prior release of QMF.

### Before you begin

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for stand-alone and requester databases.

Before you begin the installation process, complete the following steps:

- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see [“Required authorities for installing QMF”](#) on page 13.
- If you are installing Db2 QMF for z/OS, ensure that the database is Db2 for z/OS Version 9.1 New Function Mode or later. If the database release is earlier than Version 9.1 New Function Mode, upgrade the database before you continue. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).

## Procedure

To install QMF Version 12.1 in a Db2 for z/OS stand-alone or requester database where no previous release exists, complete the following steps:

1. Start the requester database into which you will be installing QMF Version 12.1.
2. Complete the steps in [“Preparing to install QMF in requester \(Db2 for z/OS\) databases” on page 33](#).  
Thoroughly test the installation of all the products that are listed as required in [“Prerequisite software” on page 33](#) before you start to install QMF. If an installation verification procedure (IVP) is associated with the product (GDDM, for example), ensure that the IVP exits with no errors before you start to install QMF.
3. Install QMF by running the jobs in [“Jobs that install QMF V12.1 where no prior release is installed” on page 45](#).
4. Customize QMF for the z/OS environment in which it is running:
  - If QMF is running in TSO, complete the steps in [“Customizing requester installations under TSO” on page 49](#).
  - If QMF is running in CICS, complete the steps in [“Customizing requester installations under CICS” on page 55](#).
5. Test the installation according to the procedure in [“Testing QMF installations on requester databases” on page 71](#).
6. Run the installation verification procedure for the appropriate environment:
  - If QMF is running in TSO, complete the steps in [“Running the IVPs for TSO” on page 80](#).
  - If QMF is running in CICS, complete the steps in [“Running the IVP for CICS” on page 82](#).
7. Thoroughly test the new release with site-specific workloads, users, applications, and objects before using it in a production environment.  
If the objects that you will use to test the new release are in a different database, you can migrate the objects to the database where QMF Version 12.1 is installed by using the instructions in [Chapter 6, “Migration and fallback considerations,” on page 85](#).

## What to do next

Continue the installation process with the following roadmaps as necessary:

- If you want to install or migrate QMF on additional stand-alone or requester databases, go back through [roadmap 1](#).
- If you want to install or migrate QMF on server databases that are to be accessed by the QMF CONNECT command, proceed to [roadmap 2](#).
- If you want to install or migrate QMF on sever databases that are to be accessed through three-part names, proceed to [roadmap 3](#).
- If you want to enable a National Language Feature, proceed to [roadmap 4](#).

## Related concepts

Initializing global variables and QMF session behavior when QMF starts

You can use several methods to set QMF global variables when QMF starts. These methods involve modifying the DSQUOPTS routine, the Q.GLOBAL\_VARS global variable table, or the default system



initialization procedure. This procedure can also be used to set other aspects of a user's QMF session before the home panel displays.

## **Installation path B: Migrating to QMF V12.1 from QMF V11.2, V11.1, V10 or a QMF NFM release in a Db2 for z/OS stand-alone or requester database**

Use this procedure to migrate QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, QMF Version 10, or QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1.

### **Before you begin**

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for stand-alone and requester databases.

Before you begin the installation process, complete the following steps:

- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see [“Required authorities for installing QMF”](#) on page 13.
- If you are installing Db2 QMF for z/OS, ensure that the database is Db2 for z/OS Version 9.1 New Function Mode or later. If the database release is earlier than Version 9.1 New Function Mode, upgrade the database before you continue. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).

### **Procedure**

To migrate to QMF Version 12.1 from a QMF New Function Mode release, from QMF Version 10.1, QMF Version 11.1, or from QMF Version 11.2 in a Db2 for z/OS requester, complete the following steps:

1. Back up the source code and any associated load modules for user-written routines or programs that are essential for your use of QMF.

Such routines could include, but are not limited to, the following items:

- Governor exit routines
- Edit codes for QMF forms
- REXX execs that provide form calculations or conditional formatting
- Applications that make use of the QMF callable or command interfaces
- Command synonym definitions and any routines that support these definitions
- Customized function key definitions

2. Start the requester database into which you will be installing QMF Version 12.1.

3. Complete the steps in [“Preparing to install QMF in requester \(Db2 for z/OS\) databases”](#) on page 33.

Thoroughly test the installation of all the products that are listed as required in [“Prerequisite software”](#) on page 33 before you start to install QMF. If an installation verification procedure (IVP) is associated with the product (GDDM, for example), ensure that the IVP exits with no errors before you start to install QMF.

4. Run the jobs in [“Jobs that migrate QMF V8 NFM, QMF V9 NFM, QMF V10, QMF V11.1, or QMF V11.2 to QMF V12.1”](#) on page 47.

5. Customize QMF for the z/OS environment in which it is running:

- If QMF is running in TSO, complete the steps in [“Customizing requester installations under TSO”](#) on page 49.
- If QMF is running in CICS, complete the steps in [“Customizing requester installations under CICS”](#) on page 55.

6. Test the installation according to the procedure in [“Testing QMF installations on requester databases”](#) on page 71.

7. Run the installation verification procedure for the appropriate environment:



- If QMF is running in TSO, complete the steps in [“Running the IVPs for TSO”](#) on page 80.
  - If QMF is running in CICS, complete the steps in [“Running the IVP for CICS”](#) on page 82.
8. Review the migration considerations in Chapter 6, [“Migration and fallback considerations,”](#) on page 85 to determine if special migration considerations for this release apply to your installation or to the functions that you will be using.
  9. Thoroughly test the new release with site-specific workloads, users, applications, and objects before using it in a production environment.

### What to do next

Continue the installation process with the following roadmaps as necessary:

- If you want to install or migrate QMF on additional stand-alone or requester databases, go back through [roadmap 1](#).
- If you want to install or migrate QMF on server databases that are to be accessed by the QMF CONNECT command, proceed to [roadmap 2](#).
- If you want to install or migrate QMF on sever databases that are to be accessed through three-part names, proceed to [roadmap 3](#).
- If you want to enable a National Language Feature, proceed to [roadmap 4](#).

### Related concepts

[Initializing global variables and QMF session behavior when QMF starts](#)

You can use several methods to set QMF global variables when QMF starts. These methods involve modifying the DSQUOPTS routine, the Q.GLOBAL\_VARS global variable table, or the default system initialization procedure. This procedure can also be used to set other aspects of a user's QMF session before the home panel displays.

## Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command

---

Use the following roadmap to identify the installation path for server-only databases that are to be accessed by the QMF CONNECT command or the DSQSDBNM program parameter. The paths in this roadmap also enable the servers to be accessed by three-part names in QMF commands.

### Restrictions:

- Connectivity with remote servers is not supported when QMF for TSO is started as a stored procedure in the Db2 for z/OS requester database.
- QMF commands that include three-part names cannot be directed to DB2 for VSE and VM servers.
- By default, QMF commands that include three-part names cannot be used to access remote tables that contain large object (LOB) data. To enable access of LOB data in remote tables with three-part names, the DSQEC\_LOB\_RETRV global variable must be set to 3. The DSQEC\_LOB\_RETRV can also be set to 2 to enable the retrieval of LOB metadata only.
- Unless the command that includes the three-part name is directed to Db2 for z/OS, QMF must be started with multirow fetch turned off.

### Before you begin

Understand the concepts in [“QMF in distributed data networks”](#) on page 5 before you use this information.

### Procedure

Use the following roadmap to identify the installation path that is appropriate for your environment.

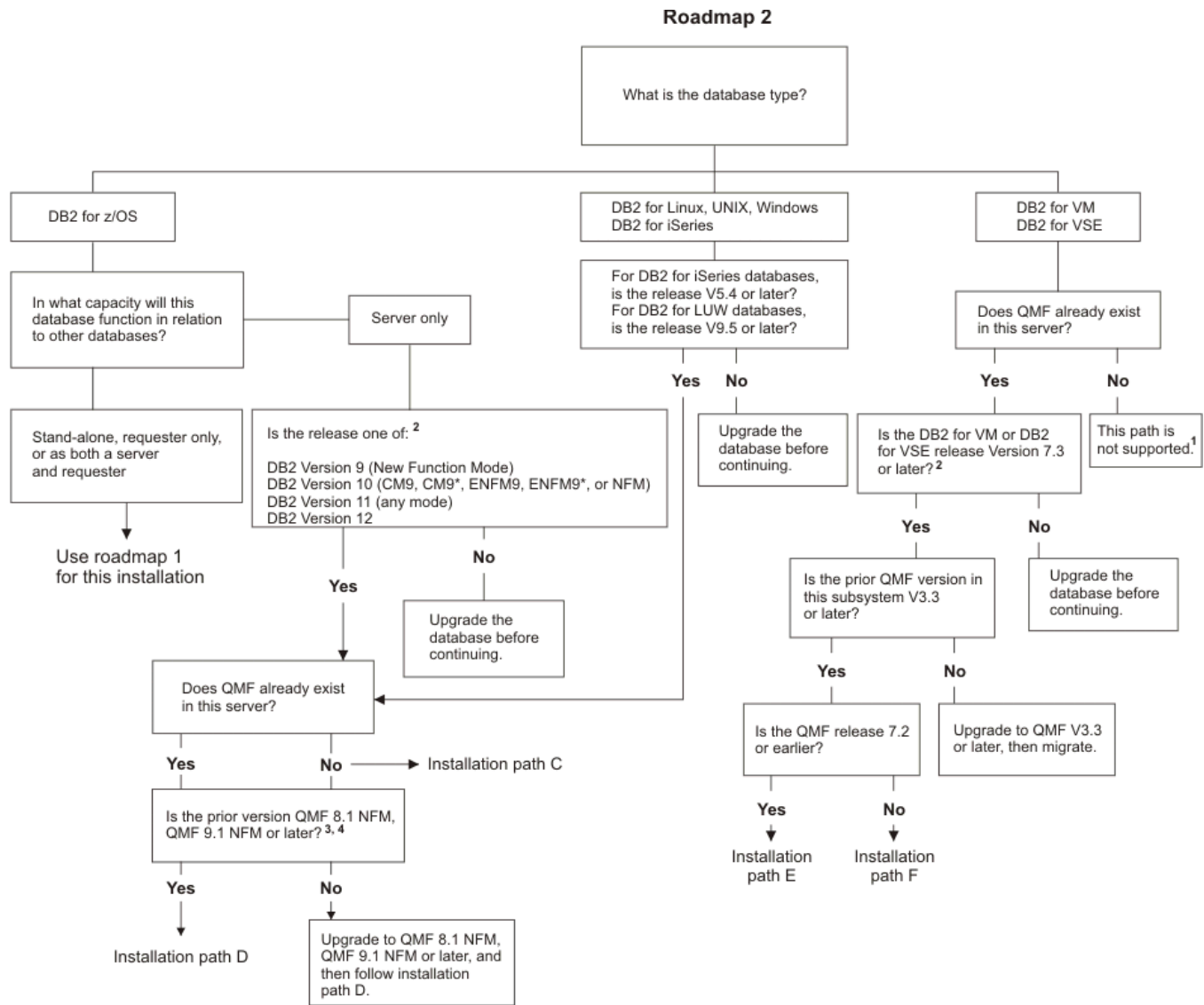


Figure 5. Installation roadmap for databases that are accessed by the QMF CONNECT command or the DSQSDBNM program parameter

**Roadmap notes:**

- a. QMF Version 12.1 does not support new installations on DB2 Server for VSE and VM. QMF Version 3.3 or later must be installed or the database cannot function as a server for requesters where QMF Version 12.1 is installed.
- b. To determine which database release is installed, complete the appropriate step:
  - For Db2 for z/OS databases, run the following query:

```
SELECT GETVARIABLE('SYSIBM.VERSION')
FROM SYSIBM.SYSDUMMY1
```

The result is DSNvrrm, where DSN indicates that this database is Db2 for z/OS, vv indicates the version, rr indicates the release level, and m indicates the modification level. For example, a Db2 for z/OS Version 8.1.5 database returns the following result:

```
DSN08015
```

- For DB2 for iSeries databases, issue the DSPPTF command. The release is recorded in the **Release of Base Option** field.

- For Db2 for Linux, UNIX, and Windows databases, run the following query:

```
SELECT SERVICE_LEVEL
FROM SYSIBMADM.ENV_INST_INFO
```

The first three numbers show the version, release, and modification level. For example, "DB2 v9.1.600.647" indicates Db2 for Linux, UNIX, and Windows Version 9.1.6.

- For DB2 for VSE and VM databases, run the following query:

```
SELECT *
FROM SYSTEM.SYSOPTIONS
WHERE SQLOPTION='RELEASE'
```

The result is *v.r.m*, where *v* indicates the version, *r* indicates the release level, and *m* indicates the modification level.

- To determine which QMF release is installed, start QMF in the local Db2 for z/OS subsystem. When the QMF home panel is displayed, issue the CONNECT command to connect to the remote server. Alternatively, you can specify the initial database connection by using the DSQSDBNM parameter when you start QMF. When you are connected to the remote server, check the version and release that is shown on the QMF home panel.

For information about the CONNECT command, see [CONNECT in CICS](#) or [CONNECT in TSO](#).

- To determine the mode of an existing QMF installation, run the following query. You can run the following query from QMF, SPUFI, or any other SQL tool that is available to you:

```
SELECT LENGTH
FROM SYSIBM.SYSCOLUMNS
WHERE NAME = 'USERID'
AND TBNAME = 'ERROR_LOG'
AND TBCREATOR = 'Q'
```

If the query returns a value of 8, the existing QMF installation is in Compatibility Mode. If the query returns a value of 128, the existing QMF installation is in New Function Mode. If the query returns no value, QMF is not installed on the server that you are checking.

## What to do next

Based on your results after you work through this roadmap, proceed to installation path C, D, E, or F.

## Related tasks

[Installation path C: Installing QMF V12.1 in a Db2 for z/OS, iSeries, or LUW server](#)

[Installation path D: Migrating to QMF V12.1 from QMF V11.2, QMF V11.1, QMF V10, or a QMF NFM release in a Db2 for z/OS, iSeries, or LUW server](#)

[Installation path E: Migrating to QMF V12.1 from V7.2 or earlier in a DB2 for VSE and VM server](#)

Use this procedure to migrate an existing release of QMF that is Version 7.2 or earlier to QMF Version 12.1 in a DB2 for VSE and VM server. This installation path prepares the DB2 for VSE and VM server for access by the QMF CONNECT command or by the DSQSDBNM program parameter.

[Installation path F: Migrating to QMF V12.1 from QMF V8.1, V9.1, V10.1, V11.1, or V11.2 in a DB2 for VSE and VM server](#)

Use this procedure to migrate an existing QMF Version 8.1, Version 9.1, Version 10.1, Version 11.1, or Version 11.2 installation to QMF Version 12.1 in a DB2 for VSE and VM server. This installation path prepares the DB2 for VSE and VM server for access by the QMF CONNECT command or by the DSQSDBNM program parameter.

[Roadmap 3: Installing server databases accessed by three-part names](#)

Use the following roadmap to identify the installation path that supports using three-part names in QMF commands to access data at remote servers.

[Specifying the initial database connection](#)

You can start QMF in the local Db2 for z/OS subsystem and connect immediately to another database before the QMF home panel is displayed.

## Installation path C: Installing QMF V12.1 in a Db2 for z/OS, iSeries, or LUW server

Use this procedure to install QMF Version 12.1 in Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows servers where no prior release of QMF is installed.

After the installation is complete, requester databases where QMF Version 12.1 is installed can connect to this server with the QMF CONNECT command or the DSQSDBNM program parameter. QMF commands that include three-part table and view names can also be directed to this server.

### Before you begin

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for server databases that are accessed by the QMF CONNECT command.

Before you begin the installation process, complete the following steps:

- If you have not done so already, install, prepare, and test QMF Version 12.1 in at least one Db2 for z/OS requester database that will access this server. Only Db2 for z/OS databases can function as requesters. See [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) for installation paths for requesters.
- Verify that the database is one of the following releases. If necessary, upgrade the database before you continue:
  - Db2 for z/OS servers must be Version 9.1 New Function Mode or later. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).
  - If the Db2 for z/OS database will function as both a requester and server, see [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) to determine your installation path; the installation paths in that roadmap enable the database to function in both capacities.
  - DB2 for iSeries servers must be Version 5.4 or later.
  - Db2 for Linux, UNIX, and Windows servers must be Version 9.5 or later.
- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see [“Required authorities for installing QMF” on page 13](#).

### Procedure

To install QMF Version 12.1 in a Db2 for z/OS, iSeries, or Linux, UNIX, and Windows server where no prior release of QMF release is installed, complete the following steps:

1. Define and test DRDA communications between the server into which you are installing QMF Version 12.1 and the requester into which you already installed QMF Version 12.1.

DRDA communications between the server and the requester are used to complete the QMF Version 12.1 installation in the server. For more information about how to define and test DRDA, see the Db2 documentation.

2. Start both the requester and server databases.

You must start the Db2 for z/OS subsystem in the requester database with distributed data facility (DDF) started.

3. For DB2 for iSeries servers only: from the DB2 for iSeries Query Manager, issue the following SQL statement on the server.

The user ID that you use must belong to the security officer or it must be a user ID that has \*ALLOBJ authority.

```
CREATE COLLECTION Q
```

4. Set installation parameters as explained in [“Setting installation job parameters for server installations” on page 61.](#)
5. Run the jobs in [“Jobs that install QMF V12.1 in server databases where no prior release is installed” on page 62.](#) You must run these jobs from the local database.
6. Test the installation in the remote server:
  - a) Complete the steps in [“Testing QMF installations on server databases” on page 83.](#)
  - b) Thoroughly test the new release with site-specific workloads, users, applications, and objects before using it in a production environment.  
 If the objects that you will use to test the new release are in a different database, you can migrate the objects to the database where QMF Version 12.1 is installed by using the instructions in [Chapter 6, “Migration and fallback considerations,” on page 85.](#)
7. For DB2 for iSeries servers only: ensure that QMF users have \*USE authority for Q \*LIB.

### What to do next

Continue the installation process with the following roadmaps as necessary:

- If you want to install or migrate QMF on additional stand-alone or requester databases, go through [roadmap 1.](#)
- If you want to install or migrate QMF on additional server databases that are to be accessed by the QMF CONNECT command, go back through [roadmap 2.](#)
- If you want to install or migrate QMF on sever databases that are to be accessed through three-part names, proceed to [roadmap 3.](#)
- If you want to enable a National Language Feature, proceed to [roadmap 4.](#)

### Related tasks

[Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command](#)

## Installation path D: Migrating to QMF V12.1 from QMF V11.2, QMF V11.1, QMF V10, or a QMF NFM release in a Db2 for z/OS, iSeries, or LUW server

Use this procedure to migrate QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, QMF Version 10, QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1 in a Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows server.

After the migration is complete, requester databases where QMF Version 12.1 is installed can connect to this server with the QMF CONNECT command or the DSQSDBNM program parameter, and QMF commands that include three-part table and view names can also be directed to this server.

### Before you begin

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for server databases that are accessed by the QMF CONNECT command.

Before you begin the installation process, complete the following steps:

- If you have not done so already, install, prepare, and test QMF Version 12.1 in at least one Db2 for z/OS requester database that will access this server. Only Db2 for z/OS databases can function as requesters. See [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) for installation paths for requesters.
- Verify that the database is one of the following releases. If necessary, upgrade the database before you continue:
  - Db2 for z/OS servers must be Version 9.1 New Function Mode or later. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).
  - DB2 for iSeries servers must be Version 5.4 or later.
  - Db2 for Linux, UNIX, and Windows servers must be Version 9.5 or later.

- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see [“Required authorities for installing QMF” on page 13.](#)

## Procedure

To migrate to QMF Version 12.1 from a QMF New Function Mode release in a Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows server, complete the following steps:

1. Define and test DRDA communications between the server into which you are installing QMF Version 12.1 and the requester into which you already installed QMF Version 12.1.  
  
DRDA communications between the server and the requester are used to complete the QMF Version 12.1 installation in the server. For more information about how to define and test DRDA, see the Db2 documentation.
2. Start both the requester and server databases.  
  
You must start the Db2 for z/OS subsystem in the requester database with distributed data facility (DDF) started.
3. Set installation parameters as explained in [“Setting installation job parameters for server installations” on page 61.](#)
4. Run the jobs in [“Jobs that migrate QMF V8 NFM, QMF V9 NFM, QMF V10, QMF V11.1, or QMF V11.2 to QMF V12.1 in a z/OS, iSeries, or LUW server” on page 64.](#) These jobs must be run from the local database.
5. Test the installation in the remote server:
  - a) Complete the steps in [“Testing QMF installations on server databases” on page 83.](#)
  - b) Thoroughly test the new release with site-specific workloads, users, applications, and objects before using it in a production environment.  
  
If the objects that you will use to test the new release are in a different database, you can migrate the objects to the database where QMF Version 12.1 is installed by using the instructions in Chapter 6, [“Migration and fallback considerations,” on page 85.](#)
6. Review the migration considerations in Chapter 6, [“Migration and fallback considerations,” on page 85](#) to determine if special migration considerations for this release apply to your installation or to the functions that you will be using.

## What to do next

Continue the installation process with the following roadmaps as necessary:

- If you want to install or migrate QMF on additional stand-alone or requester databases, go through [roadmap 1.](#)
- If you want to install or migrate QMF on additional server databases that are to be accessed by the QMF CONNECT command, go back through [roadmap 2.](#)
- If you want to install or migrate QMF on sever databases that are to be accessed through three-part names, proceed to [roadmap 3.](#)
- If you want to enable a National Language Feature, proceed to [roadmap 4.](#)

## Related tasks

[Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command](#)

## Installation path E: Migrating to QMF V12.1 from V7.2 or earlier in a DB2 for VSE and VM server

Use this procedure to migrate an existing release of QMF that is Version 7.2 or earlier to QMF Version 12.1 in a DB2 for VSE and VM server. This installation path prepares the DB2 for VSE and VM server for access by the QMF CONNECT command or by the DSQSDBNM program parameter.

### Before you begin

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for server databases that are accessed by the QMF CONNECT command.

Before you begin the installation process, complete the following steps:

- If you have not done so already, install, prepare, and test QMF Version 12.1 in at least one Db2 for z/OS requester database that will access this server. Only Db2 for z/OS databases can function as requesters. See [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) for installation paths for requesters.
- Ensure that the DB2 for VSE and VM server is Version 7.3 or later
- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see [“Required authorities for installing QMF” on page 13](#).

### Procedure

To migrate to QMF Version 12.1 in a DB2 for VSE and VM server from QMF from Version 7.2 or earlier, complete the following steps:

1. Define and test DRDA communications between the server into which you are installing QMF Version 12.1 and the requester into which you already installed QMF Version 12.1.  
  
DRDA communications between the server and the requester are used to complete the QMF Version 12.1 installation in the server. For more information about how to define and test DRDA, see the Db2 documentation.
2. Start both the requester and server databases.  
  
You must start the Db2 for z/OS subsystem in the requester database with distributed data facility (DDF) started.
3. Set installation parameters as explained in [“Setting installation job parameters for server installations” on page 61](#).
4. Run the jobs in [“Jobs that migrate QMF V7.2 or earlier to QMF V12.1 in a VM or VSE server” on page 66](#). You must run these jobs from the local database.
5. Test the installation in the remote server:
  - a) Complete the steps in [“Testing QMF installations on server databases” on page 83](#).
  - b) Thoroughly test the new release with site-specific workloads, users, applications, and objects before using it in a production environment.  
  
If the objects that you will use to test the new release are in a different database, you can migrate the objects to the database where QMF Version 12.1 is installed by using the instructions in [Chapter 6, “Migration and fallback considerations,” on page 85](#).
6. Review the migration considerations in [Chapter 6, “Migration and fallback considerations,” on page 85](#) to determine if special migration considerations for this release apply to your installation or to the functions that you will be using.

### What to do next

Continue the installation process with the following roadmaps as necessary:

- If you want to install or migrate QMF on additional stand-alone or requester databases, go through [roadmap 1](#).



- If you want to install or migrate QMF on additional server databases that are to be accessed by the QMF CONNECT command, go back through [roadmap 2](#).
- If you want to install or migrate QMF on sever databases that are to be accessed through three-part names, proceed to [roadmap 3](#).
- If you want to enable a National Language Feature, proceed to [roadmap 4](#).

### Related tasks

[Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command](#)

## Installation path F: Migrating to QMF V12.1 from QMF V8.1, V9.1, V10.1, V11.1, or V11.2 in a DB2 for VSE and VM server

Use this procedure to migrate an existing QMF Version 8.1, Version 9.1, Version 10.1, Version 11.1, or Version 11.2 installation to QMF Version 12.1 in a DB2 for VSE and VM server. This installation path prepares the DB2 for VSE and VM server for access by the QMF CONNECT command or by the DSQSDBNM program parameter.

### Before you begin

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for server databases that are accessed by the QMF CONNECT command.

Before you begin the installation process, complete the following steps:

- If you have not done so already, install, prepare, and test QMF Version 12.1 in at least one Db2 for z/OS requester database that will access this server. Only Db2 for z/OS databases can function as requesters. See “[Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)](#)” on [page 14](#) for installation paths for requesters.
- Ensure that the DB2 for VSE and VM server is Version 7.3 or later
- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see “[Required authorities for installing QMF](#)” on [page 13](#).

### Procedure

To migrate to QMF Version 12.1 in a DB2 for VSE and VM server from QMF from Version 8.1, Version 9.1, Version 10.1, Version 11.1, or Version 11.2, complete the following steps:

1. Define and test DRDA communications between the server into which you are installing QMF Version 12.1 and the requester into which you already installed QMF Version 12.1.  
  
DRDA communications between the server and the requester are used to complete the QMF Version 12.1 installation in the server. For more information about how to define and test DRDA, see the Db2 documentation.
2. Start both the requester and server databases.  
You must start the Db2 for z/OS subsystem in the requester database with distributed data facility (DDF) started.
3. Set installation parameters as explained in “[Setting installation job parameters for server installations](#)” on [page 61](#).
4. Run the jobs in “[Jobs that migrate QMF V8, V9, V10, V11.1 or V11.2 to QMF V12.1 in a VM or VSE server](#)” on [page 66](#). You must run these jobs from the local database.
5. Test the installation in the remote server:
  - a) Complete the steps in “[Testing QMF installations on server databases](#)” on [page 83](#).
  - b) Thoroughly test the new release with site-specific workloads, users, applications, and objects before using it in a production environment.  
  
If the objects that you will use to test the new release are in a different database, you can migrate the objects to the database where QMF Version 12.1 is installed by using the instructions in [Chapter 6, “Migration and fallback considerations,”](#) on [page 85](#).



6. Review the migration considerations in [Chapter 6, “Migration and fallback considerations,”](#) on page 85 to determine if special migration considerations for this release apply to your installation or to the functions that you will be using.

### **What to do next**

Continue the installation process with the following roadmaps as necessary:

- If you want to install or migrate QMF on additional stand-alone or requester databases, go through [roadmap 1](#).
- If you want to install or migrate QMF on additional server databases that are to be accessed by the QMF CONNECT command, go back through [roadmap 2](#).
- If you want to install or migrate QMF on sever databases that are to be accessed through three-part names, proceed to [roadmap 3](#).
- If you want to enable a National Language Feature, proceed to [roadmap 4](#).

### **Related tasks**

[Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command](#)

## **Roadmap 3: Installing server databases accessed by three-part names**

Use the following roadmap to identify the installation path that supports using three-part names in QMF commands to access data at remote servers.

### **Procedure**

Use the following roadmap to identify the installation path that is appropriate for your environment.

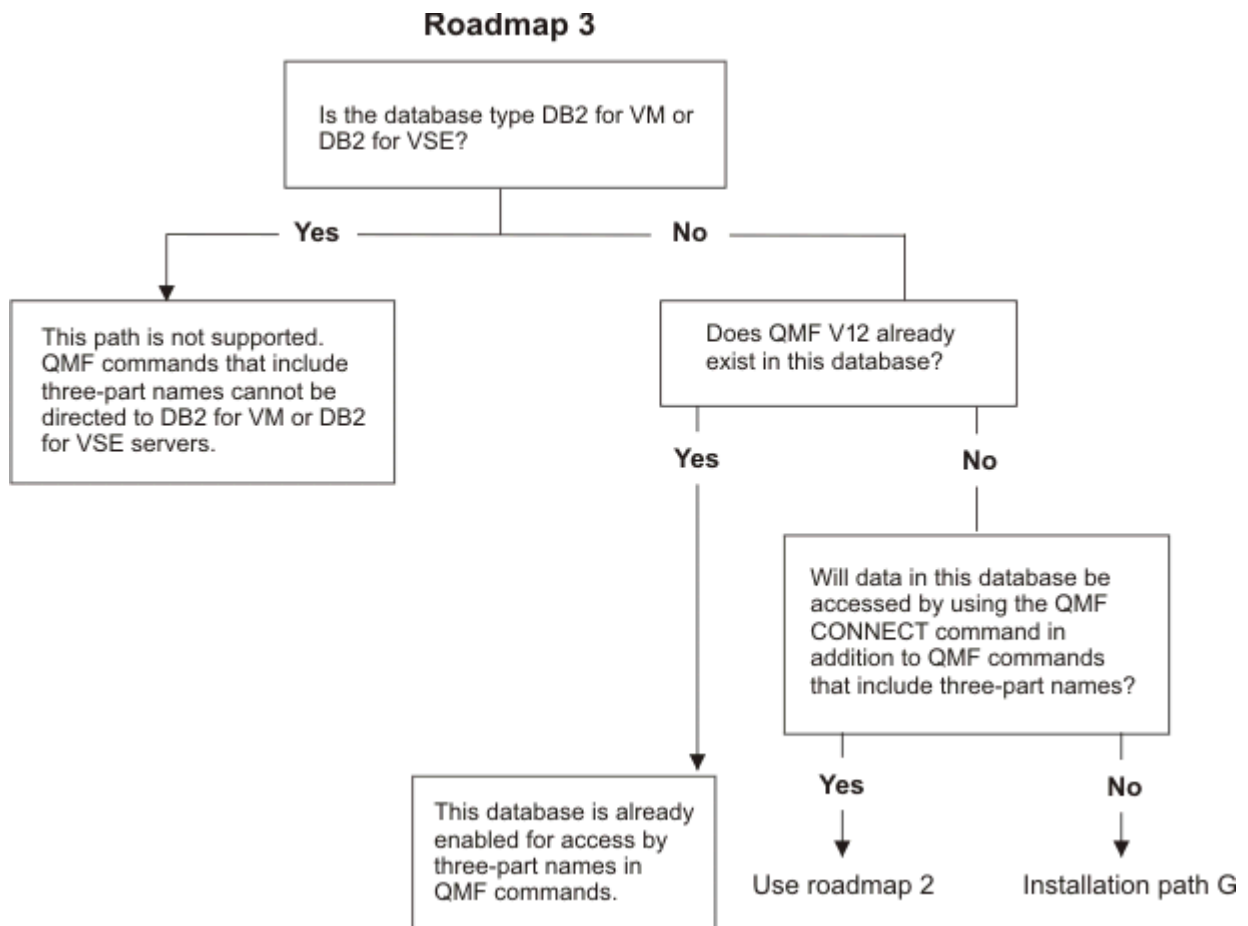


Figure 6. Installation roadmap for installations that use three-part names in QMF commands to access data at remote servers

### What to do next

Based on your results after you work through this roadmap, proceed to roadmap 2 or installation path G.

### Related tasks

[Installation path G: Preparing a remote server to be accessed by QMF commands that include three-part names](#)

[Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command](#)

## Installation path G: Preparing a remote server to be accessed by QMF commands that include three-part names

Use this procedure to prepare a remote server to be accessed by QMF commands that include three-part names.

The following rules apply to QMF commands that include three-part table and view names:

- They can be initiated from Db2 for z/OS databases only.
- They can be directed to all types of servers except DB2 for VSE and VM.
- They cannot be used when QMF for TSO has been started as a Db2 for z/OS stored procedure.

The stored procedure interface to QMF for TSO is an optional interface that allows any product that can run a Db2 for z/OS stored procedure to start QMF for TSO.

- They will produce errors if multirow fetch is active and the command is directed to a server other than Db2 for z/OS.

Start QMF in the Db2 for z/OS requester with multirow fetch turned off if users will be issuing QMF commands with three-part names to servers other than Db2 for z/OS.

By default, QMF commands that include three-part names cannot be used to access remote tables that contain large object (LOB) data. To enable access of LOB data in remote tables with three-part names, the DSQEC\_LOB\_RETRV global variable must be set to 3. The DSQEC\_LOB\_RETRV can also be set to 2 to enable the retrieval of LOB metadata only.

For more information about three-part table and view names, see the Db2 documentation.

### Before you begin

The instructions in this topic assume that you have already determined that this is the correct installation path by using the roadmap for server databases that are accessed by three-part names.

Before you begin the installation process, complete the following steps:

- If you have not done so already, install, prepare, and test QMF Version 12.1 in at least one Db2 for z/OS requester database that will access this server. Only Db2 for z/OS databases can function as requesters. See [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)” on page 14](#) for installation paths for requesters.
- Ensure that you have the proper authority for the database into which you will be installing the product. For more information, see [“Required authorities for installing QMF” on page 13](#).

### Procedure

To prepare a remote server to be accessed by QMF commands that include three-part names, complete the following steps:

1. Define and test DRDA communications between the server into which you are installing QMF Version 12.1 and the requester into which you already installed QMF Version 12.1.  
  
DRDA communications between the server and the requester are used to complete the QMF Version 12.1 installation in the server. For more information about how to define and test DRDA, see the Db2 documentation.
2. Start both the requester and server databases.  
  
You must start the Db2 for z/OS subsystem in the requester database with distributed data facility (DDF) started.
3. Run the jobs in [“Running installation jobs that prepare servers to be accessed by QMF commands that include three-part names” on page 68](#). You must run these jobs from the local database.

### Related concepts

[Enabling support for multirow fetch and insert](#)

The DSQSMRFI parameter controls whether the database uses multirow or single-row fetch and insert.

### Related tasks

[Roadmap 3: Installing server databases accessed by three-part names](#)

Use the following roadmap to identify the installation path that supports using three-part names in QMF commands to access data at remote servers.

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## Roadmap 4: Databases accessed in languages other than English

---

Use the following roadmap if you need to use QMF in languages other than English.

### Procedure

Use the following roadmap to identify the installation path that is appropriate for your environment.

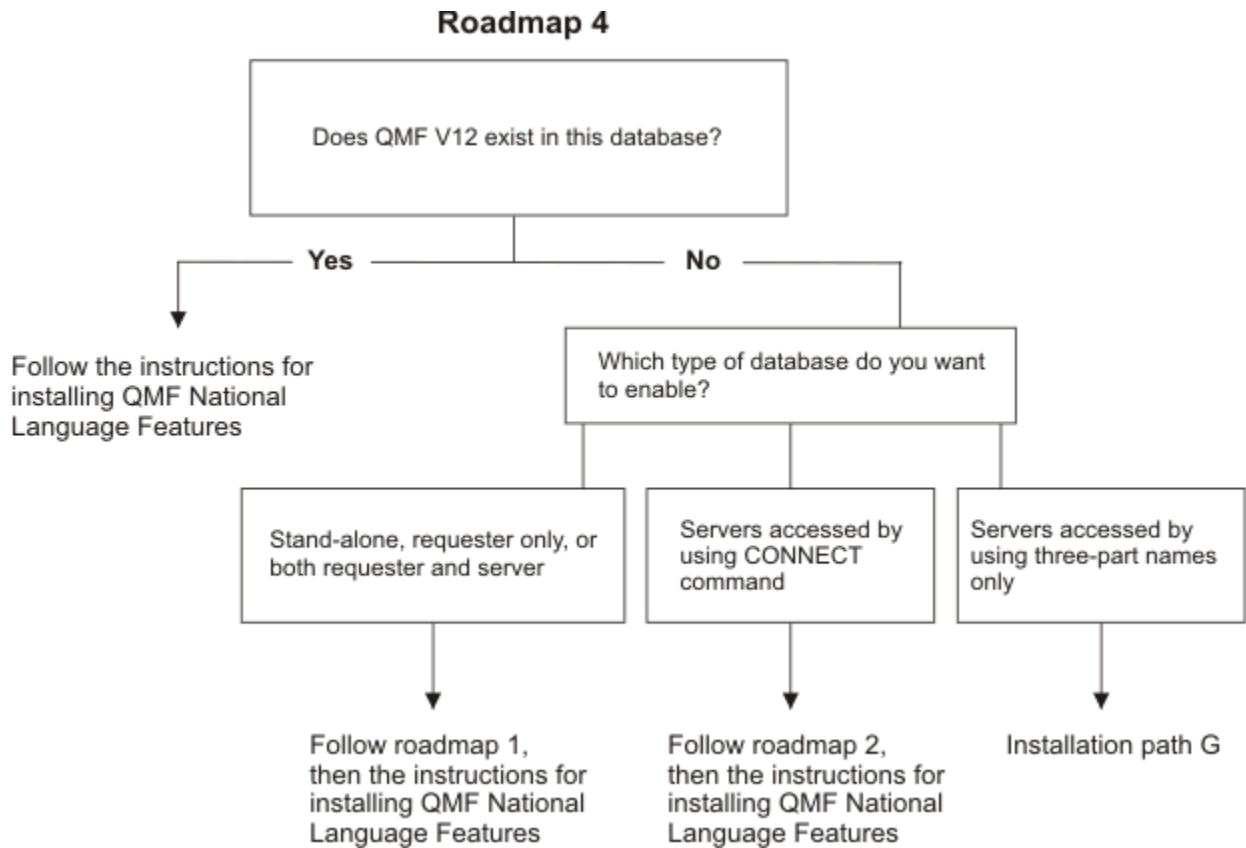


Figure 7. Installation roadmap for enabling QMF to access data in languages other than English

### What to do next

Based on your results after you work through this roadmap, proceed to roadmap 1, roadmap 2, installation path G, or the instructions for installing QMF National Language Features.

### Related tasks

Installation path G: Preparing a remote server to be accessed by QMF commands that include three-part names

Roadmap 1: Installing QMF in stand-alone or requester databases (Db2 for z/OS only)

Use this information for the first QMF installation and for databases that will stand alone, function as requesters only, or function as both requesters and servers. Only Db2 for z/OS databases can stand alone or function as requesters.

Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command

Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Tasks to perform when upgrading Db2 for z/OS but not QMF

When you upgrade either the mode or the release of a Db2 for z/OS database where QMF is installed, but you do not upgrade QMF, you must run two QMF bind jobs.

You must run these jobs even if you make no changes to the existing installation of QMF. Also, you must run these jobs regardless of the database's function in your distributed network (stand-alone, requester only, server only, or both requester and server).

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

*Table 4. Job sequence to run when you upgrade the release or mode of a Db2 for z/OS database, even if no changes are required to the existing QMF installation*

Job name	Description
DSQ1BSQL	<p>This job binds the QMF installation programs from the Db2 for z/OS requester database to the server database.</p> <p>If you installed QMF under a secondary authorization ID, edit this job and add either the OWNER(<i>secauth</i>) parameter or a SET CURRENT SQLID='<i>secauth</i>' specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.</p> <p>You must run this job first.</p>
DSQ1BPKG	<p>This job binds the QMF packages.</p> <p>You must run this job after you run the DSQ1BSQL job.</p>

**Related tasks**

Installing the enhanced LIST command function (z/OS only)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.



---

## Chapter 3. Installing or migrating QMF in requester (Db2 for z/OS) databases

Prepare your environment and system before you begin installing or migrating QMF in a Db2 for z/OS database.

---

### Preparing to install QMF in requester (Db2 for z/OS) databases

---

Prepare your environment and system before you begin installing or migrating QMF in a Db2 for z/OS database.

#### About this task

The procedures in this topic are associated with the installation paths for stand-alone or requester databases. Be sure that you are following the correct installation path before you perform these tasks.

#### What to do next

After completing these tasks, return to the installation procedure that is associated with your chosen installation path to determine the next step in the installation.

#### Related tasks

[Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)](#)

Use this information for the first QMF installation and for databases that will stand alone, function as requesters only, or function as both requesters and servers. Only Db2 for z/OS databases can stand alone or function as requesters.

### Installation prerequisites for requester (Db2 for z/OS) databases

Before you can install QMF on Db2 for z/OS databases that function as stand-alone or requester databases, you must fulfill the hardware and software requirements.

#### Prerequisite hardware

QMF runs on any processor that is supported by the operating system. However, some operations will not work with columns that contain decimal floating-point data if the processor on which QMF is running does not support decimal floating-point instructions.

QMF can access all of the DASD devices that are supported by z/OS and Db2 for z/OS, as well as all display devices supported by GDDM.

To implement a National Language Feature that uses a double-byte character set (DBCS), you need a workstation that provides DBCS support. Ensure that this device is supported by GDDM.

#### Prerequisite software

This topic lists software that is required to install and use QMF Version 12.1, as well as optional software that is necessary for special features. Each of these products might have prerequisite hardware or software that you must install and configure.

- Required software

The following table lists the program products that are required to support QMF Version 12.1.

Function	Minimum required product
Database functions	One of the following: <ul style="list-style-type: none"> <li>– Db2 for z/OS Version 9.1 (VUE and MLC) running in New Function Mode with APAR PM45482 applied</li> <li>– DB2 for z/OS Version 10.1 (VUE and MLC) running in all modes (except CM8, CM8*, ENFM8, and ENFM8*) with APARs PM50434 and PM72274 applied</li> <li>– Db2 for z/OS Version 11.1 (VUE and MLC)</li> <li>– Db2 for z/OS V12.1 (VUE and MLC)</li> </ul>
Panel display	GDDM GDDM is provided as a base element of z/OS.
QMF for CICS operations only	CICS Transaction Server for z/OS, Version 3 Release 1 or later, 5655-M15

- Optional software

GDDM-PGF Version 2.1.3 (5668-812) is required for creating charts in QMF.

### Related concepts

Required authorities for installing and administering QMF

Specific authorities are required for installing QMF and for general QMF administration.

### Related tasks

Enabling charting functions

QMF creates charts using the Interactive Chart Utility (ICU) supplied by the GDDM-PGF product.

### Related information

The IBM Publications Center  
The IBM Publications Center  
 See the documentation about the products that are required for QMF Version 12.1  
 See the documentation about the products that are required for QMF Version 12.1

<http://www.ibm.com/software/data/support/lifecycle/Search> for information about supported versions of products that are necessary for QMF Version 12.1.

<http://www.ibm.com/support/docview.wss?uid=swg21409518> See the information about compatibility between QMF Version 12.1 and other Db2 releases.

<http://www.ibm.com/support/docview.wss?uid=swg21570476> See the information about compatibility between QMF and z/OS.

## Copying the QMF Version 12.1 libraries from the distribution media

To load the QMF libraries from the distribution media to the Db2 for z/OS target system, use System Modification Program Extended (SMP/E). You copy the libraries to the system where you will be performing the first QMF Version 12.1 installation.

The QMF program directories, which are available in the Db2 Query Management Facility Library, explain how to use System Modification Program Extended (SMP/E). Ensure that you are reading the correct program directory for your release and national language, and complete all SMP/E tasks that are described in the program directory.

With SMP/E, you load QMF modules into two types of libraries:

- Target libraries, which contain the executable code of the running system

The following table describes the purpose of each target library. Library names that end with *n* have a version of the library available for each supported national language. The *n* represents a 1-character language identifier that is associated with each QMF NLF.



By default, all target libraries are prefixed with the QMF default high-level qualifier for Version 12.1, QMF1210. If you change this prefix, be sure that your new naming convention allows you to identify the version and release of QMF. This information is needed when tracing problems or communicating with IBM Software Support.

<i>Table 6. QMF Version 12.1 target libraries and their descriptions</i>	
<b>Library name</b>	<b>Description</b>
SDSQBASE	Sample base QMF for TSO and CICS SMP/E installation jobs. There is no NLF counterpart for this library.
SDSQSAM $n$	Sample NLF SMP/E installation jobs. There is no English (E) counterpart for this library.
SDSQCHRT	Chart formats supplied with QMF
SDSQCLT $n$	CLIST library
SDSQDBRM	Contains the QMF packages This library is not required for the QMF runtime environment. It is used only for installation and configuration purposes.
SDSQEXC $n$	Exec library
SDSQEXIT	Library for user exits, which can be written for: <ul style="list-style-type: none"> <li>– User-created edit codes</li> <li>– Initializing the QMF environment (the DSQUOPTS and Q.SYSTEM_INI exit routines are used for this purpose)</li> <li>– Governing resources</li> </ul> You can also use this library to store any QMF load libraries that you have customized.
SDSQLOAD	QMF load library
SDSQMAP $n$	Default GDDM maps
SDSQMLB $n$	ISPF message library, which is only required if you are running QMF under ISPF
SDSQPLB $n$	ISPF panel library, which is only required if you are running QMF under ISPF
SDSQPVR $n$	Contains non-ISPF QMF user interface panels, online help panels, and message help panels. When you run the DSQ1 $n$ PNL job during the installation process, panels from this library are used to populate the VSAM data set DSQPNL $n$ . DSQPNL $n$ is allocated to the ddname DSQPNLE when you start QMF.  This library is not required for the QMF runtime environment. It is used only for installation and configuration purposes.
SDSQSAP $n$	Contains QMF installation jobs and sample source code for both the callable interface and programs that support user-created edit codes  This library is not required for the QMF runtime environment. It is used only for installation and configuration purposes.
SDSQSLBE	ISPF skeleton library, which is required only if you are running QMF under ISPF. This library is English-only and does not have NLF counterparts.

Library name	Description
SDSQTLB $n$	ISPF table library, which is required only if you are running QMF under ISPF
SDSQUSR $n$	User library for user-written governor or user edit exit routines  This library is not required for the QMF runtime environment. It is used only for installation and configuration purposes.

- Distribution libraries, which contain the master copy of all the system elements

The following table describes the purpose of each distribution library. Library names that end with  $n$  have a version of the library available for each supported national language. The  $n$  represents a 1-character language identifier that is associated with each QMF NLF.

By default, all distribution libraries are prefixed with the QMF default high-level qualifier for Version 12.1, QMF1210. If you change this prefix, be sure that your new naming convention allows you to identify the version and release of QMF. This information is needed when tracing problems or communicating with IBM Software Support.

Library name	Description
ADSQBASE	Distribution library for sample SMP/E jobs that are used for installation
ADSQSAM $n$	Distribution library for sample SMP/E jobs that are used for NLF installation
ADSQCHRT	Distribution library supplied with QMF for GDDM charts, which reside in the SDSQCHRT library
ADSQCLT $n$	Distribution library for CLISTs, which reside in the SDSQCLT $n$ library
ADSQDBRM	Distribution library for QMF database packages, which reside in the SDSQDBRM library
ADSQEXC $n$	Distribution library for Execs, which reside in the SDSQEXC $n$ library
ADSQMAP $n$	Distribution library that is supplied with QMF for GDDM maps, which reside in the SDSQMAP $n$ library
ADSQMLB $n$	Distribution library for ISPF messages, which reside in the SDSQMLB $n$ library
ADSQOBJ	Distribution library for load modules, which are located in the SDSQLOAD library
ADSQPLB $n$	Distribution library for ISPF panels, which are located in the SDSQPLB $n$ library
ADSQPVR $n$	Distribution library for the QMF panel library, SDSQPVR $n$
ADSQSAP $n$	Distribution library for sample code, which resides in the SDSQSAP $n$ library
ADSQSLBE	Distribution library for ISPF skeletons, which reside in the SDSQSLBE library
SDSQTLB $n$	Distribution library for ISPF tables, which reside in the SDSQTLB $n$ library
SDSQUSR $n$	Distribution library for sample code that users can modify, which resides in the SDSQUSR $n$ library

The program directory provides estimates for the sizes of all target and distribution library data sets for SMP/E and QMF.

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Related information

See the information about QMF Version 12.1 program directories.

## Populating the VSAM panel library

When the SMP/E installation is complete, populate the QMF VSAM panel library, DSQPNLE.

To populate the QMF VSAM panel library, edit and run job SDSQSAPE(DSQ1EPNL). This job copies member DSQPNLE from the SDSQPVRE target library to VSAM panel data set DSQPNLE. Run this job once per SMP/E installation.

For additional information about the SMP/E installation, see the QMF program directory that is appropriate for your QMF release and national language.

## Related information

See the information about QMF Version 12.1 program directories.

## Addressing storage requirements

For the amount of storage required to copy the QMF Version 12.1 libraries from the distribution media using SMP/E, refer to the QMF program directory that is appropriate for your QMF version and national language.

When you plan your region size, consider the storage required to load modules during initialization and the virtual storage requirements for report operations.

**Restriction:** For TSO only, consider the amount of space required to run applications other than QMF.

## Related information

See the information about QMF Version 12.1 program directories.

### **Storage required to load modules during initialization**

These amounts of storage are required to load QMF modules when QMF is started. These are the required amounts if no QMF modules are placed in the link pack area.

You might improve performance of QMF operations by placing certain modules in the link pack area.

- QMF modules that must run in 24-bit addressing mode require 52 KB. Modules DSQCTOPX (used in both TSO and CICS installations) and DSQCCI (used in TSO installations only) fall into this category. They must run below 16 MB.
- QMF modules that can run in 31-bit addressing mode require 4.5 MB.

All modules except DSQCTOPX and DSQCCI can run in 31-bit addressing mode.

Almost all load modules are reentrant.

## Related concepts

### Moving modules to enhance performance

You might improve performance of QMF by placing certain modules in the link pack area. However, certain modules should not be placed in the link pack area.

### **Virtual storage requirements for report operations**

Your specific requirements might be greater than the following requirements for TSO or CICS, depending on the size of the reports your users typically create or the complexity of formatting options they use.

- Report storage in TSO

In QMF for TSO, the minimum amount of virtual storage that is required to run QMF queries and hold QMF report data is 2 MB per QMF user. You can allocate report storage in two ways in QMF for TSO:

- Use the DSQSBSTG program parameter to specify the maximum amount of storage as a fixed amount or a percentage of the available storage for the TSO region.
- Use the DSQSRSTG program parameter to specify an amount of storage to be reserved in the TSO region for applications other than QMF. QMF then dynamically allocates storage from the non-reserved amount, as needed, for report operations.

Also in QMF for TSO, LOB and XML data is stored in above-the-bar storage until it is needed for display in a report. To process QMF commands or SQL statements that involve LOB and XML data, users need access to above-the-bar storage.

- Report storage in CICS

QMF runs as a conversational transaction in CICS, where multiple users of QMF are in the same CICS address space. Allow for 16 MB of virtual storage from the extended CICS dynamic storage area (ECSDA), and add an additional 2 MB of virtual storage from this area per active QMF user. Set report storage for each user by specifying the DSQSBSTG parameter when you start QMF.

Because QMF for CICS is a large conversational transaction, consider isolating QMF transaction processing in a CICS region that is dedicated to QMF transactions. Depending on the amount of storage available, there is an upper limit on the number of users that can run QMF in the same CICS region. To support more QMF users, use multiple CICS regions by using CICS multiregion operation. You might want to route the QMF transaction from one CICS system (for example, a terminal-owning region) to the CICS system that is designated to process QMF transactions (for example, an application-owning region). If you do, use either multiple transaction IDs or dynamic transaction routing.

**Related concepts**

Defining a fixed amount of virtual storage for reports

Use the DSQSBSTG parameter to specify a fixed number of bytes of virtual storage to be used by QMF for report operations.

**Related tasks**

Reserving virtual storage from the TSO region to be used for applications other than QMF

Use the DSQSRSTG parameter in QMF for TSO to specify the maximum amount of virtual storage that you want to reserve in the TSO region for use by applications other than QMF, such as TSO commands, REXX, or ISPF. This reserved storage is then not used for QMF report operations.

**Moving modules to enhance performance**

You might improve performance of QMF by placing certain modules in the link pack area. However, certain modules should not be placed in the link pack area.

After you use SMP/E to unload the QMF libraries from the distribution media, the library QMF1210.SDSQLOAD contains the load modules for QMF.

The following table shows the modules that you can move into the link pack area to enhance performance.

<i>Table 8. Modules that can reside in the PLPA or EPLPA</i>	
<b>Module</b>	<b>Description</b>
DSQQMFE DSQQMF DSQCSUB DSQCCISW DSQCBST DSQCELTT DSQCEBLT DSQCIX DSQQMRRS DSQUOPTS DSQABA1E	These modules are used when you invoke QMF.  Modules DSQCCISW and DSQCIX are used in QMF for TSO only.

Table 8. Modules that can reside in the PLPA or EPLPA (continued)

Module	Description
DSQUEDIT DSQUECIC	These modules are related to custom exit routines that call user-defined edit codes to format data. If you expect heavy use, move them into the link pack area. DSQUEDIT is the user edit routine module for TSO, and DSQUECIC is the user edit routine module for CICS.  For more information about creating custom formatting routines that support user-defined edit codes, see <a href="#">Chapter 16, “Custom edit exit routines for QMF forms,”</a> on page 253.
DSQUEGV3	This module is the English governor module for CICS.  For more information about the governor, see <a href="#">Chapter 17, “Controlling QMF resources,”</a> on page 281.

The following table lists modules that can reside in the PLPA. These modules must run in 24-bit addressing mode.

Table 9. Modules that can reside in the PLPA

Module	Description
DSQCTOPX	QMF uses this module during initialization in both QMF for TSO and CICS.
DSQCCI	QMF uses this module for the ISPF command interface (QMF for TSO only).  For more information about this interface, see .

The following table describes the modules that should not be placed in the link pack area. Modules listed in the first three rows of the table are not reentrant.

Table 10. Modules that should not be placed in the link pack area

Module	Description
DSQCI	This module is used when QMF is invoked. It is used in TSO only.
DSQUEGV1	This module is the English governor module for TSO.  For more information about the governor, see <a href="#">Chapter 17, “Controlling QMF resources,”</a> on page 281.
DSQOBINS DSQOBSQL DSQOBTDC	These modules are QMF utilities that are used during installation and service updates. For more information about applying QMF service, see <a href="#">“Applying QMF service”</a> on page 335. These modules are used in TSO only.
DSQUXIA DSQUXIC DSQUXILE DSQUXIP	These modules are used during link-editing of the user edit routine program, DSQUEDIT (TSO) or DSQUECIC (CICS).  For more information about user edit exit routines, see <a href="#">Chapter 16, “Custom edit exit routines for QMF forms,”</a> on page 253.)

Table 10. Modules that should not be placed in the link pack area (continued)

Module	Description
DSQCIB (COBOL) DSQCICX (C/370) DSQCIA (Assembler) DSQCIFE (FORTRAN) DSQCIF (FORTRAN) DSQCIPX (PL/I) DSQCIPL (PL/I) DSQCIX (REXX)	The QMF callable interface uses these modules. They are reentrant and can be placed in the EPLPA. However, callable interface modules are small and are normally link-edited with the user's application module.
DYQCOMM DYQOCTL	QMF Analytics for TSO uses these modules.  The DYQCOMM module holds common data and command interface anchor points for QMF Analytics. The DYQOCTL module is non-executable and is used in product installation and maintenance for QMF Analytics.

## Setting installation job parameters for requester installations

With each QMF installation you run a set of batch installation jobs. You must customize the parameter and variable values that the jobs use before you run them.

### About this task

You can either change the default values in the installation defaults exec, DSQ1DEFS, or override the values in each installation job before you run it. All batch installation jobs are members of the QMF1210.SDSQSAPE data set.

### Installation defaults for common parameters

Default values in the QMF1210.SDSQEXCE(DSQ1DEFS) exec.

The following table lists the installation defaults provided by QMF for the most commonly used parameters for requester installations. Review the job comments in the DSQ1DEFS exec for explanations of additional parameters.

Table 11. Default installation values provided in the DSQ1DEFS exec		
Variable name in DSQ1DEFS exec	Default provided with QMF-	Description
SSID	DSN	Four-character name of the local Db2 for z/OS requester database into which you are installing QMF.
VCATNAME	QMFDSN	QMF table space catalog alias to be used for QMF STOGROUP creation in new installations only (not migrations).
VOLUMES	'*'	Value of the VOLUMES parameter to be used for STOGROUP creation in new installations only (not migrations).
QMFLPLNAME	QMF1210	Name of the QMF application plan.
DBPTGROUP	0	The DBPARTITIONNUM value on the CREATE DATABASE PARTITION GROUP statement for installation on Db2 UDB servers.
DBCCSID	UNICODE	The default CCSID value for the CREATE DATABASE statement on a Db2 for z/OS server.
DSQWLMNM		The default WLM environment that is used for defining QMF stored procedures and user-defined functions.

Table 11. Default installation values provided in the DSQ1DEFS exec (continued)

Variable name in DSQ1DEFS exec	Default provided with QMF-	Description
SECAUTH	No default	If your site uses RACF® security groups and you therefore must install QMF in a Db2 for z/OS database (whether local or remote) under a secondary authorization ID, use this variable to specify that ID. The secondary authorization ID that you specify must have SYSADM or equivalent authority for the target database into which QMF is being installed.  If you do not provide a value for this variable, QMF does not try to process a secondary authorization ID for the steps of the installation that bind packages and plans.
EXTSEC	" "	Specification of internal or external Db2 security control. The default is internal Db2 security control. If a value of YESEXSEC is found, no GRANT statements will be issued in QMF installation jobs. In addition, the default list views used for tables defined in job QMF1210.SDSQSAPE(DSQ1BVW) will not reference the SYSIBM.SYSTABAUTH catalog table. The option does not affect the enhanced list views created in QMF1210.SDSQSAPE(DSQ1BUDV).
(See <a href="#">Note 1</a> for the following variables.)		
TSCT1PRI TSCT1SEC	200 20	The DSQTSTCT1 table space holds the Q.OBJECT_DIRECTORY table.
TSCT2PRI TSCT2SEC	200 20	The DSQTSTCT2 table space holds the Q.OBJECT_REMARKS table.
TSCT3PRI TSCT3SEC	5000 200	The DSQTSTCT3 table space holds the Q.OBJECT_DATA and Q.OBJECT_DATA2 tables.
TSPROPRI TSPROSEC	100 20	The DSQTSPRO table space holds the Q.PROFILES table.
TSLOGPRI TSLOGSEC	100 20	The DSQTSLOG table space holds the Q.ERROR_LOG table.
TSSYNPRI TSSYNSEC	100 20	The DSQTSSYN table space holds the Q.COMMAND_SYNONYMS table.
TSGOVPRI TSGOVSEC	100 20	The DSQTSGOV table space holds the Q.RESOURCE_TABLE table.
TSRDOPRI TSRDOSEC	12 4	The DSQTSRDO table space holds the Q.DSQ_RESERVED table.

Table 11. Default installation values provided in the DSQ1DEFS exec (continued)

Variable name in DSQ1DEFS exec	Default provided with QMF-	Description
TSDEFPRI TSDEFSEC	100 20	The DSQTSDEF table space holds tables that result from the QMF SAVE DATA command. Optional job DSQ1STGJ creates this table space and uses these default values.
TSGLVPRI TSGLVSEC TSGLVBP TBGLVVARVALUE	100 20 BP0 2000	The DSQTSGLV table space holds the Q.GLOBAL_VARS table. The TSGLVPRI and TSGLVSEC parameters specify the default primary and secondary space allocations for DSQTSGLV. TSGLVBP specifies the default buffer pool for DSQTSGLV. TBGLVVARVALUE specifies the default VARCHAR length of the VARVALUE column in Q.GLOBAL_VARS.
(See Note 2 for the following variables.)		
IXODRPRI IXODRSEC	200 20	These variables specify space quantities for Q.OBJECT_DIRECTORYX, the index for the Q.OBJECT_DIRECTORY table.
IXORMPRI IXORMSEC	200 20	These variables specify space quantities for Q.OBJECT_REMARKSX, the index for the Q.OBJECT_REMARKS table.
IXODTPRI IXODTSEC	200 20	These variables specify space quantities for Q.OBJECT_OBJDATA, the index for the Q.OBJECT_DATA table.
IXOPROPRI IXOPROSEC	200 20	These variables specify space quantities for Q.PROFILEX, the index for the Q.PROFILES table.
IXCOMPRI IXCOMSEC	100 20	These variables specify space quantities for Q.COMMAND_SYNONYMSX, the index for the Q.COMMAND_SYNONYMS table.
IXGLVPRI IXGLVSEC	100 20	These variables specify space quantities for Q.GLOBAL_VARSX, the index for the Q.GLOBAL_VARS table.

**Notes:**

1. The variables in this section specify default primary and secondary space allocations (shown in 1 KB units) for the QMF control tables. The QMF control tables are installed only if no prior release of QMF exists in the database. Variables with PRI in the name denote the primary space allocation; variables with SEC in the name denote the secondary space allocation. The values of these variables are not commonly changed.
2. The variables in this section specify default primary and secondary space allocations (in 1 KB units) for the indexes on the QMF control tables. Indexes are created only when no prior release of QMF exists in the database.

Not all QMF installation jobs use every value in the DSQ1DEFS exec. The installation jobs clearly describe each DSQ1DEFS value that is referenced; if a DSQ1DEFS value is not referenced in a job, it is ignored.

**Related concepts**

[Overriding defaults in the installation jobs](#)



An example JCL that shows how to override the inherited default values within an installation job by modifying the parameter values on the SYSTSIN statement.

### Related tasks

[Setting site-specific installation defaults for requester installations in the exec](#)  
Change the default installation values by modifying the installation defaults exec.

### Related reference

[QMF control tables and table spaces for TSO and CICS](#)

These are the control tables shipped with QMF.

### Setting site-specific installation defaults for requester installations in the exec

Change the default installation values by modifying the installation defaults exec.

### About this task

The defaults for most of the database installation values in each installation job are in the QMF1210.SDSQEXCE(DSQ1DEFS) exec. The installation jobs inherit default values from this exec, so you change the values to those that are appropriate for your site and for your installation. Not all installation parameters are found in the DSQ1DEFS exec, so are not inherited by the installation job. For those parameters you override the values in each job if you do not want to use the default values.

**Important:** Do not delete variables from DSQ1DEFS.

### Procedure

To modify any of the values in the DSQ1DEFS exec, complete the following steps:

**Tip:** You can complete similar steps to modify a copy of DSQ1DEFS to use for each database installation and then use the copied and renamed file to provide defaults for all QMF installation jobs for that database.

1. Copy QMF1210.SDSQEXCE(DSQ1DEFS) to any member name.
2. Edit the new member and change the default values shown in [Table 11 on page 40](#) to your site-specific values.  
For example, the following sample lines specify a new default value of DB2L for SSID, ZOS1DB2L for LOCATION, and DB2LDSN for VCATNAME:

```
SSID = "DB2L"  
LOCATION = "ZOS1DB2L"  
VCATNAME = "DB2LDSN"
```

3. Modify the DSQDEFS DD statement for the job to point to the copy of DSQ1DEFS that you modified.
4. Remove the variables from the SYSTSIN statement in the job. Be sure to leave the QMFBSQL value after the DSQ1INST call.

### Example

The following example shows lines from job DSQ1BLNM, which shows installation defaults for QMF Version 12.1 (QMF1210) with Db2 for z/OS Version 11.1 (DSN1110). If you are installing QMF Version 12.1 with a Db2 for z/OS version other than Version 11.1, use the qualifier for that version. For example, the default Db2 for z/OS Version 11 high-level qualifier is DSN1110.

```
//DSQ1BLNM JOBcard  
//DSQEXSQL PROC RGN='2048K',  
//          QMFTPRE='QMF1210',  
//          DB2EXIT='DSN1110.SDSNEXIT',  
//          DB2LOAD=DSN1110.SDSNLOAD'  
//STEP1 EXEC PGM=IKJEFT01,REGION=&RGN  
//STEPLIB DD DSN=&QMFTPRE.SDSQLOAD,DISP=SHR  
//          DD DSN=&DB2EXIT.,DISP=SHR  
//          DD DSN=&DB2LOAD.,DISP=SHR  
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=132  
//SYSTEM DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*
```

```

//SYSEXEC DD DSN=&QMFTPRE..SDSQEXCE,DISP=SHR
//DSQDEFS DD DSN=&QMFTPRE..SDSQEXCE(DSQ1DEFS),DISP=SHR
//DSQINDD DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BNFM),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLND),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNA),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNR),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNO),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNE),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLC),DISP=SHR
// PEND
//DSQCTBL EXEC DSQEXSQL
//* =====
//* Tailor SSID and LOCATION values below:
//* =====
//STEP1.SYSTSIN DD *
%DSQ1INST QMFBSQL <-- Variables have been removed after this line

```

## Related concepts

[Installation defaults for common parameters](#)

Default values in the QMF1210.SDSQEXCE(DSQ1DEFS) exec.

[Overriding defaults in the installation jobs](#)

An example JCL that shows how to override the inherited default values within an installation job by modifying the parameter values on the SYSTSIN statement.

### Overriding defaults in the installation jobs

An example JCL that shows how to override the inherited default values within an installation job by modifying the parameter values on the SYSTSIN statement.

The QMF installation jobs inherit values from the DSQ1DEFS exec. Each job is liberally commented with instructions about the values that you must customize and how to submit the job.

In this example the following default values are changed:

- The ssid default value has been changed to DB2L.
- The location default value has been changed to ZOS1DB2L.
- The vcatname default value has been changed to DB2LDSN.

```

//DSQ1BLNM JOBcard
//DSQEXSQL PROC RGN='2048K',
//          QMFTPRE='QMF1210',
//          DB2EXIT=DSN1110.SDSNEXIT',
//          DB2LOAD=DSN1110.SDSNLOAD'
//STEP1 EXEC PGM=IKJEFT01,REGION=&RGN
//STEPLIB DD DSN=&QMFTPRE..SDSQLOAD,DISP=SHR
// DD DSN=&DB2EXIT.,DISP=SHR
// DD DSN=&DB2LOAD.,DISP=SHR
//SYSTSPRT DD SYSOUT=*,DCB=BLKSIZE=132
//SYSTEM DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSEXEC DD DSN=&QMFTPRE..SDSQEXCE,DISP=SHR
//DSQDEFS DD DSN=&QMFTPRE..SDSQEXCE(DSQ1DEFS),DISP=SHR
//DSQINDD DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BNFM),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLND),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNA),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNR),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNO),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1BLNE),DISP=SHR
// DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLC),DISP=SHR
// PEND
//DSQCTBL EXEC DSQEXSQL
//* =====
//* Tailor SSID and LOCATION values below:
//* =====
//STEP1.SYSTSIN DD*
%DSQ1INST QMFBSQL SSID(DB2L) LOCATION(ZOS1DB2L) +
          VCATNAME(DB2LDSN) VOLUMES('*')

```

Not all installation parameters are found in the DSQ1DEFS exec; therefore, not all values can be inherited by each installation job. The following additional parameters are used by the QMF installation jobs and must be specified within each job before it is run if you do not want to use the default values:

Table 12. Installation job parameters not found in the installation defaults file, DSQ1DEFS

Installation variable	Default value	Description
QMFTPRE	QMF1210	QMF target library prefix
DB2EXIT	DSN1210.SDSNEXIT	Local Db2 for z/OS exit library The default value points to the Db2 for z/OS Version 12 exit library. If you are installing QMF Version 12.1 with a Db2 for z/OS version other than Version 12, use the qualifier for that version. For example, the default Db2 for z/OS Version 9 high-level qualifier is DSN910.
DB2LOAD	DSN1210.SDSNLOAD	Local Db2 for z/OS load library The default value points to the Db2 for z/OS Version 12 exit library. If you are installing QMF Version 12.1 with a Db2 for z/OS version other than Version 12, use the qualifier for that version. For example, the default Db2 for z/OS Version 9 high-level qualifier is DSN910.

After completing the tasks associated with preparing for installation, return to the installation procedure that is associated with your chosen installation path to determine the next step in the installation.

### Related concepts

[Installation defaults for common parameters](#)

Default values in the QMF1210.SDSQEXCE(DSQ1DEFS) exec.

### Related tasks

[Setting site-specific installation defaults for requester installations in the exec](#)

Change the default installation values by modifying the installation defaults exec.

## Running the installation jobs for requester (Db2 for z/OS) databases

You must run installation jobs to install or migrate to QMF Version 12.1 in a Db2 for z/OS requester database.

### Jobs that install QMF V12.1 where no prior release is installed

This installation path installs QMF Version 12.1 in a Db2 for z/OS requester database when no prior release of QMF is installed.

#### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. The procedures in this topic are associated with Installation path A: Installing QMF V12.1 in a Db2 for z/OS requester.
- Ensure that the database is Db2 for z/OS Version 9.1 New Function Mode or later. If the database release is earlier than Version 9.1 New Function Mode, upgrade the database before you continue. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-

new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).

### About this task

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

### Procedure

- Customize parameter values in the jobs as necessary by following the instructions in the prolog of each job or by setting site-specific installation defaults or by overriding installation defaults. Then run the jobs in their specified order.

The jobs listed in the following table reside as members in the QMF1210.SDSQSAPE data set.

<i>Table 13. Job sequence for installing QMF Version 12.1 in a Db2 for z/OS Version 9 New Function Mode (or later) database when no prior release of QMF exists</i>	
<b>Job name</b>	<b>Description</b>
DSQ1TBAJ	Optional: Creates QMF VCAT name.
DSQ1BSQL	Binds the QMF installation programs to the Db2 for z/OS requester database.  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID= <i>secauth</i> specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.
DSQ1BLNI	Creates QMF Version 12.1 control tables.
DSQ1BVW	Creates the default views that support the QMF LIST command.
DSQ1BPKG	Binds the QMF application packages.
DSQ1BINR	Binds the QMF application plan.  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID= <i>secauth</i> specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.
DSQ1STGJ	Creates the storage group, database, and table space needed for the SAVE DATA command. This command is used in the QMF installation verification procedure (IVP).  Before running this job, ensure that the TSDEFPRI and TSDEFSEC variables are properly set for this installation. For more information, see <a href="#">“Setting installation job parameters for requester installations”</a> on page 40.
DSQ1EIVS	Optional: Creates the QMF sample tables..

- After you run each job, check for a return code of 0 or 4. Both codes indicate successful completion.

## What to do next

After all jobs have run successfully, return to “Installation path A: Installing QMF V12.1 in a Db2 for z/OS stand-alone or requester database” on page 16 for the next step in the installation process.

## Related tasks

Installing the enhanced LIST command function (z/OS only)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.

## Jobs that migrate QMF V8 NFM, QMF V9 NFM, QMF V10, QMF V11.1, or QMF V11.2 to QMF V12.1

This installation path migrates a release of QMF New Function Mode, QMF Version 10, QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1 in a Db2 for z/OS database that will function as a requester or as both a requester and server.

### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. The procedures in this topic are associated with Installation path B: Migrating to QMF V12.1 from QMF V11.2, QMF V11.1, QMF V10, or a QMF NFM release in a Db2 for z/OS requester.
- Ensure that the existing release of QMF is Version 8.1 New Function Mode, Version 9.1 New Function Mode, Version 10, Version 11.1, or Version 11.2.
- Ensure that the database is Db2 for z/OS Version 9.1 New Function Mode or later. If the database release is earlier than Version 9.1 New Function Mode, upgrade the database before you continue. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).

### About this task

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

### Procedure

- Customize parameter values in the jobs as necessary by following the instructions in the prolog of each job or by setting site-specific installation defaults or by overriding installation defaults. Then run the jobs in their specified order.

The jobs listed in the following table reside as members in the QMF1210.SDSQSAPE data set.

Job name	Description
DSQ1BSQL	Binds the QMF installation programs to the Db2 for z/OS requester database.  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID= <i>secauth</i> specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID.

Table 14. Job sequence to migrate from QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, QMF Version 10, QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1 (continued)

Job name	Description
DSQ1BGLV	Creates new QMF Version 12.1 Q.GLOBAL_VARS control table. <b>Exception:</b> You do not need to run DSQ1BGLV if you are migrating from QMF 11.1 or higher.
DSQ1ERTS	Inserts new command synonyms RUNTSO and RU into Q.COMMAND_SYNONYMS and updates the DPREF definition.
DSQ1BVW	Creates the default QMF views that support the QMF LIST command. If you installed the enhanced LIST command function, run the DSQ1BUDV installation job after you run DSQ1BVW.
DSQ1BPKG	Binds the QMF packages. In this job, the DSQDEFS DD statement is optional.
DSQ1BINR	Binds the QMF application plan in the requester database.  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID= <i>'secauth'</i> specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.

- After you run each job, check for a return code of 0 or 4. Both codes indicate successful completion.
- Run these jobs if you want the sample tables from a previous release of QMF.  
Previous releases of the QMF sample tables are still valid in the new QMF version.
  - If the older sample tables were not previously installed and you want to install them with the new version, run job DSQ1EIVS.
  - If the older sample tables were previously installed and you want to reinstall them, first run job DSQ1EDSJ to delete the existing samples. Then run job DSQ1EIVS.

### What to do next

After all jobs have run successfully, return to “Installation path B: Migrating to QMF V12.1 from QMF V11.2, V11.1, V10 or a QMF NFM release in a Db2 for z/OS stand-alone or requester database” on page 18 for the next step in the installation process.

### Related tasks

[Setting installation job parameters for requester installations](#)

With each QMF installation you run a set of batch installation jobs. You must customize the parameter and variable values that the jobs use before you run them.

[Installing the enhanced LIST command function \(z/OS only\)](#)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists

that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.

## Defining programs, resources, and preferences to TSO and CICS

---

After installation, you must define and allocate certain resources and customize preferences in supporting products and components such as ISPF and GDDM. The required customization depends on the environment in which QMF will be running.

### About this task

**Important:** None of the QMF libraries require APF authorization for any function that QMF performs in either TSO or CICS.

## Customizing requester installations under TSO

If you are running QMF for TSO, you must customize it after installation. You configure GDDM and prepare the logon procedure.

### Customizing GDDM external defaults

If you are running QMF for TSO, you must customize some external defaults to ensure that GDDM correctly displays information from the database.

### About this task

QMF receives information from and presents information to the screen through services that are provided by GDDM.

To ensure that the data that is sent by Db2 is displayed properly by GDDM, the GDDM code page must match the coded character set identifiers (CCSIDs) for the current database manager. A CCSID contains all of the information necessary to assign and preserve the meaning and rendering of characters through various stages of processing and interchange. This information always includes at least one code page. The CCSID also has an associated encoding scheme that governs how various code points are to be handled.

QMF compares the GDDM application code page (APPCPG) with two different CCSID values:

- CURRENT APPLICATION ENCODING SCHEME special register

The application encoding scheme for the database is specified in this special register in Db2 for z/OS. For single-byte character set (SBCS) languages, the GDDM application code page should match the value of this special register.

- SYSIBM.SYSTEM\_EBCDIC\_CCSID session variable

This session variable specifies the CCSID that is in use for EBCDIC data on the current system. For double-byte character set (DBCS) languages, the GDDM application code page should match the value of this session variable.

If the CCSIDs do not match, there could be discrepancies between what is sent to Db2 for display and what is displayed by GDDM.

If necessary, revise the GDDM APPCPG parameter value in one of the following ways:

- Modify the GDDM external defaults file, ADMADFT, then assemble and link-edit the file. Ensure that this file is accessible to QMF during initialization.
- Create and allocate a data set that contains the GDDM defaults, as shown here.

### Procedure

This example shows how to change the APPCPG parameter to a value of 037, which is compatible with the CCSIDs in use on Db2 for z/OS:



1. Create a fixed-block sequential file with a record length of 80 named QMF1210.ADMDEFS.

QMF1210 is the default prefix for the QMF Version 12.1 libraries. If you changed the default prefix for the QMF libraries, substitute that prefix in place of QMF1210.

2. Open the ADMDEFS file for editing and insert the following ADMMDFT statement to specify a value of 037 for the APPCPG parameter. Insert the statement that starts in column 2.

```
ADMMDFT APPCPG=037
```

If you insert the line that starts in column 1, it is automatically considered a comment.

3. Save the file and add the following DD statement to the JCL that you use to start QMF:

```
//ADMDEFS DD DSN=QMF1210.ADMDEFS,DISP=SHR
```

If you changed the default prefix for the QMF libraries, substitute that prefix in place of QMF1210. See [“Preparing the TSO logon procedure”](#) on page 50 for more information about allocating data sets in TSO.

You can also use the following TSO command to allocate the file:

```
ALLOC FI(ADMDEFS) DS(QMF1210.ADMDEFS) SHR
```

## What to do next

In addition to changing the value of the APPCPG parameter, make sure that the code page of the terminal emulator that you are using is consistent with the other CCSIDs in use.

If your users need to display or print the euro currency symbol, ensure that the value that you assign to the APPCPG parameter is associated with the code page for this symbol. For example, to display the euro currency symbol in English output, the proper APPCPG value to use is 1140. Determine the appropriate APPCPG value to use for the national language in which you are running QMF and assign that value to the APPCPG parameter.

## Related information

Search for information about how to modify the external defaults file in the GDDM documentation.

## Checking that QMF devices are queryable

GDDM must be able to query device characteristics for all devices used in QMF operations.

QMF manages the display of panels and data by using GDDM services. To build and convert each data stream into a device-specific form, GDDM must be able to query device characteristics for all devices used in QMF operations.

To make sure that each device can be queried, check the bytes of the PSERVIC operand of the VTAM® MODEENT macro. These bytes define the display device or printer type and are set differently for queryable versus nonqueryable devices.

## Related information

Search for information about the valid values of the PSERVIC operand in the GDDM documentation.

## Preparing the TSO logon procedure

You must modify the TSO logon procedure to support the storage and other requirements of your users. This step includes allocating load libraries and data sets.

## About this task

The Terminal Monitor Program (TMP) is the principal interface between the user and the display device during the user's TSO sessions. Your site might be using either its own TMP or the standard one that is supplied by IBM. If the TMP is not the standard one, some of the information in this topic might not apply.

Whenever you log on to TSO, the TMP invokes the TSO logon procedure. The TSO logon procedure allocates resources for its users at the start of a TSO session. QMF users require more resources than TSO



users who are not using QMF. By using a logon procedure, you ensure that you are providing these additional resources to establish an adequate TSO environment.

IBM supplies a sample logon procedure named DSQ1EINV that you can use and modify, if necessary. The sample logon procedure allocates resources for someone who uses TSO solely as a means to reach QMF. For users who want to do more with their TSO sessions, additional resources might be required. Some of the resources that are allocated in the logon procedure can also be allocated in a CLIST or REXX exec that invokes QMF.

To establish resources for QMF operations under TSO, modify the sample TSO logon procedure as follows:

**Procedure**

1. Edit QMF1210.SDSQSAPE(DSQ1EINV).
2. Locate the REGION parameter and ensure that it meets the minimum storage requirements as described in “Addressing storage requirements” on page 37.  
For example:

```
//DSQ1EINV EXEC PGM=IKJEFT01,TIME=1440,DYNAMNBR=30,REGION=4096K
```

3. Allocate program load libraries.

Program load libraries for ISPF, ISPF/PDF, QMF, Db2 for z/OS, and GDDM must be available from the STEPLIB statement or through a CLIST before you start QMF. This step lists the load libraries for the various products and shows example allocation statements.

- a) Determine whether you want to allocate the program modules through the STEPLIB statement or through a CLIST. Add the QMF user exit library, QMF1210.SDSQEXIT, to the STEPLIB concatenation if needed. This step is required only if any exits reside in QMF1210.SDSQEXIT. The sample logon procedure includes the load libraries for ISPF, ISPF/PDF, QMF, Db2 for z/OS, and GDDM. Not all of these libraries need to be included on the STEPLIB statement. Some can be allocated later through a CLIST. Before you start QMF, a CLIST can allocate the ISPF and QMF libraries as ISPLLIB data sets.

The following figure shows the STEPLIB statement for these load libraries in the DSQ1EINV logon procedure:

```

//*****
//*                PROGRAM LOAD LIBRARIES                *
//*****
//STEPLIB DD DSN=QMF1210.SDSQEXIT,DISP=SHR          * QMF MODULES *
//          DD DSN=QMF1210.SDSQLOAD,DISP=SHR        * QMF MODULES *
//          DD DSN=ISP.SISPLOAD,DISP=SHR            * ISPF MODULES *
//          DD DSN=DSN1110.SDSNEXIT,DISP=SHR        * DB2 MODULES *
//          DD DSN=DSN1110.SDSNLOAD,DISP=SHR        * DB2 MODULES *
//          DD DSN=GDDM.SADMMOD,DISP=SHR            * GDDM MODULES *

```

*Figure 8. Load libraries in the DSQ1EINV logon procedure*

You can also use an ISPLLIB DD statement rather than the STEPLIB statement to allocate the ISPF load libraries.

- b) Determine whether you want to run coexisting versions of QMF in the same Db2 for z/OS subsystem.

If two different QMF versions will coexist, you must use separate logon procedures to allocate the QMF load libraries for each release.

The following table shows the load module library names for QMF Version 12.1 and prior QMF releases from which migration is supported.

QMF version	Load module library name
Version 12 Release 1.0	QMF1210.SDSQLOAD

QMF version	Load module library name
Version 11 Release 2.0	QMF1120.SDSQLOAD
Version 11 Release 1.0	QMF1110.SDSQLOAD
Version 10 Release 1.0	QMF1010.SDSQLOAD
Version 9 Release 1.0	QMF910.SDSQLOAD
Version 8 Release 1.0	QMF810.SDSQLOAD
Version 7 Release 2.0	QMF720.SDSQLOAD
Version 7 Release 1.0	QMF710.SDSQLOAD
Version 6 Release 1.0	QMF610.SDSQLOAD
Version 3 Release 3.0	QMF330.DSQLOAD

4. Allocate SDSQEXCE to either SYSEXEC or SYSPROC.

Allocation to SYSEXEC is shown in the following figure. The SYSPROC, SYSEXEC, and SYSHELP allocation statements show default data set names, which might be different in your installation.

Use the DD statement that has been established by your installation for the TSO search order for programs. This search order is affected by settings in the TSO defaults modules IRXTSPRM and IRXISPRM, the TSO EXECUTIL command, and the TSO ALTLIB command. If you do not know your installation's search order for REXX programs, allocate SDSQEXCE to both SYSEXEC and SYSPROC.

```

//*****
//*          DATA SETS USED BY TSO          *
//*****
//SYSPROC DD DSN=SYS2.CLIST,DISP=SHR
//          DD DSN=QMF1210.SDSQCLTE,DISP=SHR
//SYSEXEC DD DSN=SYS2.EXEC,DISP=SHR
//          DD DSN=QMF1210.SDSQEXCE,DISP=SHR
//SYSHELP DD DSN=SYS1.HELP,DISP=SHR
//EDT     DD DSN=&EDIT,UNIT=SYSDA,SPACE=(1688,(40,12))
//UTL     DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(TRK,(10,5))
//SYSPRINT DD TERM=TS,SYSOUT=A
//SYSTEM  DD TERM=TS,SYSOUT=A
//SYSIN   DD TERM=TS

```

Figure 9. Allocating SDSQEXCE in the logon procedure

5. Customize ISPF libraries, if appropriate.

ISPF libraries are optional. If you start QMF under ISPF or use ISPF-related functions, allocate these libraries as shown in the following figure. The names shown in the figure are the default names of the ISPF libraries; these names might be different for your specific ISPF installation.

```

//*****
//*          DATASETS USED BY ISPF          *
//*****
//ISPLLIB DD DSN=QMF1210.SDSQPLBE,DISP=SHR
//          DD DSN=ISP.SISPPENU,DISP=SHR
//ISPLLIB DD DSN=QMF1210.SDSQMLBE,DISP=SHR
//          DD DSN=ISP.SISPMENU,DISP=SHR
//ISPLLIB DD DSN=QMF1210.SDSQSLBE,DISP=SHR
//          DD DSN=ISP.SISPSENU,DISP=SHR
//          DD DSN=ISP.SISPLIB,DISP=SHR
//ISPTLIB DD DSN=QMF1210.SDSQTLBE,DISP=SHR
//          DD DSN=ISP.SISPTENU,DISP=SHR
//ISPPROF DD UNIT=SYSDA,SPACE=(TRK,(9,1,4)),
//          DCB=(LRECL=80,BLKSIZE=8800,RECFM=FB,DSORG=PO)

```

Figure 10. Allocating ISPF libraries in the logon procedure

6. Verify GDDM data sets.

The following table lists the name and purpose of each GDDM data set:

<i>Table 15. GDDM data sets</i>	
<b>Data set name</b>	<b>Description</b>
ADMGGMAP	GDDM map group for panels mapped by QMF The default map library is QMF1210.SDSQMAPE.
ADMCFORM	Chart formats supplied by QMF; use only if charting functions are required The default chart library is QMF1210.SDSQCHRT.
DSQUCFRM	User-defined chart objects Users who have created their own chart formats while using GDDM-PGF under QMF for TSO can retrieve chart objects from this data set or save chart objects to it.
ADMSYMBL	Symbol sets supplied with GDDM; use only if charting functions are required
ADMGDF	Graphics Data Format files; use only if charting functions are required
ADMCDATA	Chart data files; use only if charting functions are required
ADMDEFS	Allocate the ddname ADMDEFS if you modified this file to include any settings for GDDM external defaults, such as the default for the APPCPG parameter. The ADMDEFS file also stores GDDM nicknames if you use GDDM services for printing.

The following figure shows the allocation statements for these data sets in the DSQ1EINV procedure.

```
//*****
//*          QMF/GDDM DATA SETS          *
//*****
//ADMGGMAP DD DSN=QMF1210.SDSQMAPE,DISP=SHR * GDDM Map Group
//ADMCFORM DD DSN=QMF1210.SDSQCHRT,DISP=SHR * QMF-Supplied Chart Formats
//DSQUCFRM DD DSN=aaaaaaaa,DISP=SHR          * Saves User-defined ICUFORMS
//ADMCDATA DD DSN=xxxxx,DISP=SHR
//ADMGDF   DD DSN=xxxxx,DISP=SHR
//ADMSYMBL DD DSN=xxxxx,DISP=SHR
```

Figure 11. Allocation statements for GDDM data sets

To allocate these data sets in your own procedure, complete the following steps:

- Ensure that the ADMGGMAP DD statement points to the QMF ADMGGMAP library, QMF1210.SDSQMAPE, as shown here.
- Allocate separate libraries for users who want to save their own chart formats that they have created while running GDDM-PGF under QMF for TSO.  
Create the new library with a DD statement like the one in the following example. Provide values for the DSN, UNIT, VOL, and SPACE parameters, but do not change the DCB parameters.

```
//DSQUCFRM DD DSN=aaaaaaaa,DISP=(NEW,CATLG),
//          UNIT=xxxxx,VOL=SER=yyyy,
//          SPACE=(400,(200,50,25)),
```

```
//          DCB=(LRECL=400, BLKSIZE=400, RECFM=F)
```

- 1) Locate the entry for DSQUCFRM in DSQ1EINV.
  - 2) Duplicate and customize this entry for each data set that is allocated to DSQUCFRM.
  - 3) In each duplicated entry, replace *aaaaaaaa* with the name of each data set that is allocated to DSQUCFRM.
- c) Replace *xxxx* in the DD statements for ADMCDATA, ADMGDF, and ADMSYMBL with the name of the data set created during GDDM installation.  
If these data sets do not exist, define them using statements like these:

```
//ADMCDATA DD DSN=xxxx, DISP=(NEW, CATLG),
// UNIT=xxxx, SPACE=(TRK, (5, 1, 10)),
// DCB=(RECFM=F, LRECL=400, BLKSIZE=400, DSORG=PO)
```

#### 7. Allocate QMF data sets.

This table lists data sets that are used by QMF in TSO. These files are allocated to ddnames that begin with DSQ. If you want to allocate them differently, be sure to change them in the allocation statements in the logon procedure, CLIST, or exec that you use to invoke QMF.

<i>Table 16. Data sets used by QMF in TSO</i>	
<b>Data set</b>	<b>Description</b>
DSQPNLE	QMF VSAM panel file  This file changes from one release to the next and is built when you run install job QMF1210.SDSQSAPE(DSQ1EPNL).
DSQDUMP	QMF snap dump output
DSQDEBUG	QMF trace dump output  For more information about this data set, see <a href="#">“The trace facility” on page 356</a> .
DSQPRINT	Print data output
DSQSPILL	Spill data file  Instead of allocating a file for spill data, you can spill data to extended storage. For more information, see <a href="#">“Spilling report data to extended virtual storage (TSO only)” on page 152</a> .
DSQEDIT	Edit transfer file  This file temporarily holds a query or procedure that is referenced on an EDIT command while ISPF is called for editing services.

Data sets DSQDEBUG and DSQDUMP default to a printer. You can customize the definition to send the information to a data set instead.

The DD statements for DSQDUMP, DSQDEBUG, and DSQPRINT all require a DCB parameter. For DSQPRINT, add 1 to the LRECL value for the print control character.

The following figure shows the default allocation statements for the DSQ group of data sets.

```

//*****
//*          DATA SETS USED BY QMF          *
//*****
//DSQPNLE DD DSN=QMF1210.DSQPNLE,DISP=SHR
//DSQPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//DSQDEBUG DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//DSQEDIT DD UNIT=SYSVIO,DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029),
// DISP=NEW,SPACE=(CYL,(1,1))
//DSQDUMP DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//SYSUDUMP DD SYSOUT=A
//DSQSPILL DD DSN=QMF1210.DSQSPILL,DISP=(NEW,DELETE),
// UNIT=SYSVIO,SPACE=(CYL,(10,20),RLSE),
// DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)

```

Figure 12. Default allocation statements for QMF DSQ data sets

### What to do next

After completing these tasks, return to the installation procedure that is associated with your chosen installation path to determine the next step in the installation.

### Related concepts

#### [Creating QMF user profiles](#)

All QMF users need access to a user profile, which determines how QMF handles individual input from specific users. Use the profile to control certain aspects of a user's environment, such as where printer output is routed or whether input is converted to upper case.

#### [Setting program parameters and preferences at startup time](#)

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

#### [Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

#### [Format of the CALL statement](#)

This topic explains the format of the required CALL statement.

### Related tasks

#### [Using GDDM services to handle printing](#)

You can use GDDM (rather than QMF) services to handle printing in native z/OS batch, TSO, ISPF, and CICS.

### Related reference

#### [Coexistence of releases](#)

QMF Version 12.1 can coexist in the same database only with QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, or QMF Version 10.1, QMF Version 11.1, or QMF Version 11.2.

#### [Installing sample queries and procedures required for the installation verification procedures](#)

Install the sample QMF queries and procedures only on TSO.

### Related information

Search for information about GDDM data sets and external defaults in the GDDM documentation.

## Customizing requester installations under CICS

Before customizing QMF for CICS, you must install and customize both Db2 for z/OS and GDDM to run in the CICS environment.

### Related information

See the GDDM documentation.

## Describing QMF to Db2

Before you run QMF under CICS, you must describe QMF to Db2.

### Procedure

1. Install the Db2 for z/OS-to-CICS connection and the Db2 for z/OS attachment facility for CICS.

QMF uses the CICS/Db2 attachment facility to access Db2 for z/OS data in the CICS environment.

2. Verify that a DB2CONN definition exists for the database in which QMF is installed.

You can use the CICS Resource Definition Online (RDO) facility to create the definition. The DB2ENTRY definition specific to QMF will be created when the DSQ1ECSJ job is run.

All QMF programs were bound during installation; you do not need to separately bind for CICS.

## Describing QMF to CICS

Before you run QMF under CICS, you must describe QMF to CICS by modifying and running a job that updates the CICS system definitions (CSDs).

### About this task

Job DSQ1ECSJ creates a new list called QMF, which is defined in the CICS system definition file (CSD). CICS offers a utility program (DFHCSDUP) to update the CSD with a batch job. Use DFHCSDUP to update all QMF/CICS control tables except the DCT. DSQ1ECSJ also defines the DB2ENTRY that is associated with the correct plan name and group by the QMF transaction.

### Procedure

1. Use the RDO VIEW `Lsrpool name` command to check the current definitions of the LSRPOOL.

The QMF panel data set (DSQPNLE) requires a VSAM CI size of 32 K. QMF does not explicitly define an LSRPOOL entry. Instead, QMF defaults to the CICS default of 1. If the LSRPOOL in your installation is smaller than 32 KB, use DFHCSDUP to specify an LSRPOOL that supports a VSAM CI size of 32 KB.

2. Edit QMF1210.SDSQSAPE(DSQ1ECSJ) and verify or change the installation parameters in the instream procedure of the job to accommodate your site's needs.

For example:

```
//DSQ1ECSJ PROC REG=2048K,      Job step region
//      QMFTPRE='QMF1210',      DSN prefix for QMF
//      CLOAD='CICS.SDFHLOAD',   Name of CICS program library
//      CCSD='CICS.DFHCSJ',     Name of CICS CSD file
//      OUTC='*'                Print sysout class
```

3. Submit the job and check that the job ran with a return code of 0. If you receive a non-zero return code, check the job output and correct the error.

## Preparing the governor exit for CICS

Before QMF can be started under CICS, you must translate, assemble, and link-edit the governor supplied with QMF.

### Procedure

1. Edit job QMF1210.SDSQSAPE(DSQ1EGLK) according to the comments in the job.
2. Submit job QMF1210.SDSQSAPE(DSQ1EGLK).
3. Check for a return code of 0 on all steps except LKEDPROG, which can have a return code of 4. If the return code is not 0 or 4, correct the problem and rerun the job.

## Related concepts

[Controlling QMF resources](#)

A governor exit routine helps you limit end-user activity and control use of system resources at your site.

### **Customizing for GDDM support under CICS**

Additional customization is required to support GDDM under CICS.

#### ***Customizing GDDM external defaults***

You must set several GDDM external defaults in the ADMADFC file to ensure proper interaction with QMF

You must set the following external defaults:

#### **IOSYNCH**

Ensure that the IOSYNCH external default is set to YES.

#### **APPCPG**

QMF receives information from and presents information to the screen through services that are provided by GDDM. To ensure that the data that is sent by Db2 is displayed properly by GDDM, the GDDM code page must match the coded character set identifiers (CCSIDs) for the current database manager. A CCSID contains all of the information necessary to assign and preserve the meaning and rendering of characters through various stages of processing and interchange. This information always includes at least one code page. The CCSID also has an associated encoding scheme that governs how various code points are to be handled.

QMF compares the GDDM application code page (APPCPG) with two different CCSID values:

- CURRENT APPLICATION ENCODING SCHEME special register

The application encoding scheme for the database is specified in this special register in Db2 for z/OS. For single-byte character set (SBCS) languages, the GDDM application code page should match the value of this special register.

- SYSIBM.SYSTEM\_EBCDIC\_CCSID session variable

This session variable specifies the CCSID that is in use for EBCDIC data on the current system. For double-byte character set (DBCS) languages, the GDDM application code page should match the value of this session variable.

If the CCSIDs do not match, there could be discrepancies between what is sent to Db2 for display and what is actually displayed by GDDM. If necessary, revise the GDDM APPCPG parameter value in the GDDM external defaults file, ADMADFC. An APPCPG value of 037 is compatible with the CCSIDs in use on Db2 for z/OS. When you are finished modifying the file, assemble and link-edit the file and ensure that it is accessible to QMF during initialization. For details on how to modify external defaults, see the GDDM documentation in [the IBM Publications Center](#).

In addition to changing the value of the APPCPG parameter, make sure that the code page of the terminal emulator that you are using is consistent with the other CCSIDs in use.

If your users need to display or print the euro currency symbol, ensure that the value that you assign to the APPCPG parameter is associated with the code page for this symbol. For example, to display the euro currency symbol in English output, the proper APPCPG value to use is 1140. Determine the appropriate APPCPG value to use for the national language in which you are running QMF and assign that value to the APPCPG parameter.

#### ***Checking that QMF devices are queryable***

GDDM must be able to query device characteristics for all devices used in QMF operations.

QMF manages the display of panels and data by using GDDM services. To build and convert each data stream into a device-specific form, GDDM must be able to query device characteristics for all devices used in QMF operations.

To make sure that each device can be queried, check the bytes of the PSERVIC operand of the VTAM MODEENT macro. These bytes define the display device or printer type and are set differently for queryable versus nonqueryable devices.

#### **Related information**

Search for information about the valid values of the PSERVIC operand in the GDDM documentation.

### Loading QMF GDDM maps to the GDDM ADMF data set

This procedure replaces the existing maps in the data set.

#### About this task

The ADMF data set can contain maps from only one QMF release at a time.

#### Procedure

To load the GDDM maps defined with QMF Version 12.1 to the GDDM ADMF data set, follow these steps:

1. Edit QMF1210.SDSQSAPE(DSQ1EADM) and verify that the installation parameters in the instream procedure of the job, as well as the job steps, accommodate your requirements.

For example:

```
//DSQ1EADM PROC RGN='2048K',      Job-step region size
//          QMFTPRES='QMF1210',   QMF prefix name for target libraries
//          GDDMADM='GDDM.ADMF'   GDDM ADMF data set name
```

2. Submit QMF1210.SDSQSAPE(DSQ1EADM).

If you must revert to the maps from the release of QMF from which you are migrating for any reason, run job QMFvrm.SDSQSAPE(DSQ1EADM), where *vrm* is the version, release level, and modification level of the release from which you migrated.

3. Check for a return code of 0. If the return code is not 0, correct the problem and rerun DSQ1EADM.

### Creating sample charts and the QMF trace data set

This job runs statements (in DSQ1CFRM) that create the default QMF chart formats and the QMF trace data set.

#### Before you begin

If you migrated to QMF Version 12.1 from a prior QMF release, skip this step.

#### Procedure

1. Edit QMF1210.SDSQSAPE(DSQ1BFRM).
2. Locate the installation parameters in the instream procedure of the job and ensure that their values accommodate your requirements.

For example:

```
//DSQ1BFRM PROC QMFTPRES='QMF1210',  DSN prefix for QMF product
//          GDDMADM='GDDM.ADMF',      GDDM ADMF data set name
//          CHRTVOL='QMFVOL',         QMF/GDDM charts volume
//          TRCVOL='QMFVOL',         Trace data set volume
```

3. Edit DSQ1CFRM COPY, which is referenced on the SYSIN statement of the DSQ1BFRM job.
4. Customize the VSAM control statement for your installation.

For example:

```
DEFINE CLUSTER (NAME(QMF1210.DSQCFCRM) -
                VOLUMES(QMFVOL) -
                UNIQUE -
                RECSZ(400 400) -
                CONTROLINTERVALSIZE(2048) -
                KEYS(20 0)) -
                DATA -
                (RECORDS(1000 300)) -
                CATALOG(VSAMUSERCAT)
```

5. Submit QMF1210.SDSQSAPE(DSQ1BFRM).
6. Check for a return code of 0. If the return code is not 0:
  - a) Edit DSQ1CFRM and remove the steps that ran successfully; otherwise, you will receive error messages indicating that the objects are already there.



- b) Check the trace data set, DSQDEBUG, for errors. See [“The trace facility”](#) on page 356 for more information about this data set.
- c) Correct any problems you find and rerun the job.

### Updating the CICS startup job stream

You must update the DD statements in the CICS startup job stream to ensure that the proper data sets are accessed during QMF initialization.

#### Procedure

1. Place the load library that contains QMF, GDDM, and Db2 for z/OS modules in the CICS module load library list, DFHRPL.

For example:

```
//DFHRPL DD ...
//      DD DSN=QMF1210.SDSQEXIT,DISP=SHR
//      DD DSN=QMF1210.SDSQLOAD,DISP=SHR
//      DD DSN=GDDM.SADMMOD,DISP=SHR
//      DD DSN=DSN1110.SDSNEXIT,DISP=SHR
//      DD DSN=DSN1110.SDSNLOAD,DISP=SHR
```

Be sure that the correct Db2 for z/OS release level is specified on the statements that refer to the Db2 exit and load libraries.

2. Provide access to the following data sets, which are required by GDDM and QMF:

```
//*      GDDM DATA SETS
//ADMF   DD DSN=GDDM.ADMF,DISP=SHR      QMF map group
//ADML   DD SYSOUT=A
//ADMS   DD SYSOUT=A
//ADMT   DD SYSOUT=A
//*      QMF DATA SETS
//DSQPNLE DD DSN=QMF1210.DSQPNLE,DISP=SHR  QMF panel file
//DSQDEBUG DD DSN=QMF1210.DSQDEBUG,DISP=SHR Trace and error messages
//DSQUCFRM DD DSN=QMF1210.DSQUCFRM,DISP=SHR User-defined ICU forms
```

3. Shut down and restart CICS to incorporate your changes to the CICS tables and to the CICS startup job.

### Determining the type of storage to use for EXPORT and IMPORT commands

After QMF Version 12.1 is installed, the default use of CICS temporary storage and transient data queues is active. However, for compatibility reasons, QMF Version 12.1 still allows you to activate support of TSO data sets.

#### About this task

Prior releases of QMF allowed the direct use of TSO data sets from QMF transactions. However, using TSO data sets can cause unpredictable results in CICS address spaces that are running QMF transactions. Therefore, you should use CICS temporary storage or transient data queues for QMF EXPORT and IMPORT commands.

#### Procedure

To activate support for TSO data sets for QMF IMPORT and EXPORT commands, follow these steps:

1. Disable the QMF export/import control module, DSQCTLXI.

To disable this module, use the CEMT transaction supplied with CICS:

```
CEMT SET PROGRAM(DSQCTLXI) DISABLE
```

DSQCTLXI can also be disabled by removing it from the CICS CSD file. After you disable DSQCTLXI, all QMF sessions running in CICS use TSO data sets for EXPORT and IMPORT commands.

2. Set the execution key of QMF module DSQCBST to a value of CICS if you are using CICS storage protection (SIT STGPROT=YES).

To determine if storage protection is being used, issue the following command:

```
CEMT INQUIRE SYSTEM
```

If the STOREPROTECT option is set to ACTIVE, storage protection is being used.

If the EXECKey option in the DSQCBST module is set to USER, complete these steps to change this value to CICS:

a) Issue the following command:

```
CEDA ALTER PROGRAM(DSQCBST)
```

b) Recycle the CICS region.

c) Issue the following command:

```
CEMT INQUIRE PROGRAM(DSQCBST)
```

With your cursor next to the Prog option, press Enter to display all options. Check that the EXECKey option is set to Cexeckey, which indicates a value of CICS.

### What to do next

After support for CICS temporary storage or transient data queues is disabled, you can reactivate the support by issuing a CEMT command or by adding a program entry to the CICS CSD file if the entry was removed. To use CEMT, enter the following command:

```
CEMT SET PROGRAM(DSQCTLXI) ENABLE
```

Use the IMPORT command sparingly in CICS, because it can affect QMF performance for other users in the same address space. QMF uses GET/PUT services when operating under QSAM, which can lock out other QMF users in the same CICS region during I/O operations.

### Revising the EDSA limit size to accommodate SQL queries up to 2 MB

QMF Version 12.1 supports a query size of up to 2 MB for SQL queries that are directed to Db2 for z/OS.

### About this task

The DSQEC\_SQLQRYSZ\_2M parameter defines the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above the 16 MB boundary. An EDSALIMIT value of at least 62914560 is required to run SQL queries up to 2 MB in QMF for CICS.

### Procedure

- If you plan to run queries of this size, set the DSQEC\_SQLQRYSZ\_2M global variable to 1 and revise the value of the CICS EDSALIM parameter.

### What to do next

After completing these tasks, return to the installation procedure that is associated with your chosen installation path to determine the next step in the installation.

---

## Chapter 4. Installing or migrating QMF in server databases

This topic explains how to install or migrate to QMF Version 12.1 in a remote server. A remote server is a server that is accessible, through DRDA communications, from your local Db2 for z/OS subsystem.

### Before you begin

Before you install or migrate to QMF Version 12.1 in a remote server, see “Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command” on page 19 to verify that you are following the correct installation path for your required configuration.

### About this task

The installation tasks for remote servers vary depending on whether you intend to access the remote server by using remote unit of work or by using distributed unit of work.

### Related concepts

[QMF in distributed data networks](#)

The first installation of QMF must be performed in a Db2 for z/OS database. When this installation is complete, the Db2 for z/OS database can then stand alone, function as a requester, or function as both a requester and server for other QMF Version 12.1 installations.

---

## Setting installation job parameters for server installations

With each installation of QMF you run a set of batch installation jobs. You must customize the parameter and variable values that the jobs use before you run them.

### About this task

You can either change the default values in the installation defaults exec, DSQ1DEFS, or override the values in each installation job before you run it. All batch installation jobs are members of the QMF1210.SDSQSAPE data set.

### Restrictions:

- QMF commands that include three-part names cannot be directed to DB2 for VSE and VM servers.
- By default, QMF commands that include three-part names cannot be used to access remote tables that contain large object (LOB) data. To enable access of LOB data in remote tables with three-part names, the DSQEC\_LOB\_RETRV global variable must be set to 3. The DSQEC\_LOB\_RETRV can also be set to 2 to enable the retrieval of LOB metadata only.
- Connectivity with remote servers is not supported when QMF for TSO is started as a stored procedure in the Db2 for z/OS requester database.

### Procedure

To customize the parameter and variable values that the installation uses, follow one of these steps:

- Change the default values in the installation defaults exec, DSQ1DEFS.

In addition to the installation defaults listed for common parameters, QMF uses the following parameters during server installs:

#### LOCATION

This parameter is used for installations in servers that will be accessed by the CONNECT command or the DSQSDBNM program parameter. Replace the default value, THISLOCN, with the name of the remote server into which you are installing QMF.

### T3PARTNM

This parameter is associated with Installation path I: Preparing a remote server to be accessed by QMF commands that include three-part names.

Specify YES for this parameter if data will be accessed at the remote server using only three-part names in QMF commands. If you intend to use both the QMF CONNECT command and commands that include three-part names to access data at the remote server, leave this parameter blank, because the installation path for servers accessed by the CONNECT command automatically provides three-part-name access.

Review the job comments in the DSQ1DEFS exec for more information about the installation parameters.

- Override the default values by customizing the values in each job before you run it.

**Important:** Be sure that the SSID parameter in all installation jobs that are directed to remote servers is set to the ID of the local Db2 for z/OS subsystem from which you will be performing the remote installation.

### What to do next

After completing these tasks, return to the installation procedure that is associated with your chosen installation path to determine the next step in the installation.

### Related concepts

[Installation defaults for common parameters](#)

Default values in the QMF1210.SDSQEXCE(DSQ1DEFS) exec.

[Overriding defaults in the installation jobs](#)

An example JCL that shows how to override the inherited default values within an installation job by modifying the parameter values on the SYSTSIN statement.

## Running installation jobs that prepare servers for access by the QMF CONNECT command

---

You must run installation jobs to install or migrate to QMF Version 12.1 in a server database.

Connectivity with remote servers is not supported when QMF for TSO is started as a stored procedure in the Db2 for z/OS requester database..

Choose the topic appropriate for the type of installation you are performing:

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## Jobs that install QMF V12.1 in server databases where no prior release is installed

This job sequence installs QMF Version 12.1 on Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows servers where no prior release of QMF is installed.

### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. This procedure is associated with Installation path D: Installing QMF V12.1 in a Db2 for z/OS, iSeries, or LUW server.
- Verify that the database is one of the following releases. If necessary, upgrade the database before you continue:

- Db2 for z/OS servers must be Version 9.1 New Function Mode or later. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).
- DB2 for iSeries servers must be Version 5.4 or later.
- Db2 for Linux, UNIX, and Windows servers must be Version 9.5 or later.

### About this task

**Restriction:** DB2 for VSE and VM do not support new installations of QMF Version 12.1 of any type. To prepare a DB2 for VSE and VM server for access by the QMF CONNECT command, see one of the following installation paths:

- [“Installation path E: Migrating to QMF V12.1 from V7.2 or earlier in a DB2 for VSE and VM server” on page 25](#)
- [“Installation path F: Migrating to QMF V12.1 from QMF V8.1, V9.1, V10.1, V11.1, or V11.2 in a DB2 for VSE and VM server” on page 26](#)

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

### Procedure

- Customize parameter values in the jobs as necessary by following the instructions in the prolog of each job or by setting site-specific installation defaults or by overriding installation defaults. Then run the jobs in their specified order.

The jobs listed in the following table reside as members in the QMF1210.SDSQSAPE data set.

Job name	Description
DSQ1TBAJ	Optional: Creates QMF VCAT (Db2 for z/OS servers only)
DSQ1BSQL	Binds the QMF installation programs from the Db2 for z/OS requester database to the server database  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID= <i>'secauth'</i> specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.
DSQ1BLNI	Creates QMF Version 12.1 control tables
DSQ1BVW	Creates the default views that support the QMF LIST command
DSQ1BPKG	Binds the QMF packages
DSQ1STGJ	Db2 for z/OS servers only: creates the storage group, database, and table space for the QMF SAVE DATA command  The SAVE DATA command is used during the installation verification procedures.
DSQ1EIVS	Optional: Db2 for z/OS servers only: creates the QMF sample tables

Table 17. Job sequence for installing QMF Version 12.1 in remote Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows servers where no prior release of QMF is installed. (continued)

Job name	Description
DSQ1EDJ4	Optional: Db2 for Linux, UNIX, and Windows servers only: creates the QMF sample tables
DSQ1EAS4	Optional: DB2 for iSeries only: creates the QMF sample tables

- After you run each job, check for a return code of 0 or 4. Both codes indicate successful completion.

### What to do next

After all jobs have run successfully, return to [“Installation path C: Installing QMF V12.1 in a Db2 for z/OS, iSeries, or LUW server”](#) on page 22 for the next step in the installation process.

### Related tasks

[Installing the enhanced LIST command function \(z/OS only\)](#)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.

## Jobs that migrate QMF V8 NFM, QMF V9 NFM, QMF V10, QMF V11.1, or QMF V11.2 to QMF V12.1 in a z/OS, iSeries, or LUW server

This job sequence migrates a release of QMF New Function Mode, QMF Version 10, QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1.

### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. This procedure is associated with Installation path E: Migrating to QMF V12.1 from a QMF V11.2, V11.1, V10, or QMF NFM release in a Db2 for z/OS, iSeries, or LUW server.
- Verify that the database is one of the following releases. If necessary, upgrade the database before you continue:
  - Db2 for z/OS servers must be Version 9.1 New Function Mode or later. DB2 10 cannot be running in conversion mode from Version 8, conversion mode\* from Version 8, enabling-new-function mode from Version 8, or enabling-new-function mode\* from Version 8 (CM8, CM8\*, ENFM8, or ENFM8\*).
  - DB2 for iSeries servers must be Version 5.4 or later.
  - Db2 for Linux, UNIX, and Windows servers must be Version 9.5 or later.

### About this task

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

### Procedure

- Customize parameter values in the jobs as necessary by following the instructions in the prolog of each job or by setting site-specific installation defaults or by overriding installation defaults. Then run the jobs in their specified order.

The jobs listed in the following table reside as members in the QMF1210.SDSQSAPE data set.

Table 18. Job sequence to migrate from QMF Version 8.1 New Function Mode, QMF Version 9.1 New Function Mode, QMF Version 10, QMF Version 11.1, or QMF Version 11.2 to QMF Version 12.1 in a Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows server

Job name	Description
DSQ1BSQL	Binds the QMF installation programs from the Db2 for z/OS requester database to the server database  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID='secauth' specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.
DSQ1BGLV	Creates new QMF Version 12.1 Q.GLOBAL_VARS control table.  <b>Exception:</b> You do not need to run DSQ1BGLV if you are migrating from QMF 11.1 or higher.
DSQ1ERTS	Inserts new command synonyms RUNTSO and RU into Q.COMMAND_SYNONYMS and updates the DPRE definition.
DSQ1BVV	Creates the default QMF views that support the QMF LIST command. If you installed the enhanced LIST command function, run the DSQ1BUDV installation job after you run DSQ1BVV.
DSQ1BPKG	Binds the QMF packages

- After you run each job, check for a return code of 0 or 4. Both codes indicate successful completion.
- Run these jobs if you want the sample tables from a previous release of QMF. Previous releases of the QMF sample tables are still valid in the new version.
- If the older sample tables were not installed and you want to install them with the new version, run the appropriate job from the this table:

Table 19. Jobs that install sample tables	
Job name	Description
DSQ1EIVS	Installs sample tables on Db2 for z/OS servers
DSQ1EAS4	Installs sample tables on DB2 for iSeries servers
DSQ1EDJ4	Installs sample tables on Db2 for Linux, UNIX, and Windows servers

- If the older sample tables were previously installed and you want to reinstall them, first delete the existing samples.
  - Job DSQ1EDSJ deletes these samples from Db2 for z/OS databases
  - Job DSQ1EDX2 deletes them from DB2 for iSeries and Db2 for Linux, UNIX, and Windows databases.

### What to do next

After all jobs have run successfully, return to [“Installation path D: Migrating to QMF V12.1 from QMF V11.2, QMF V11.1, QMF V10, or a QMF NFM release in a Db2 for z/OS, iSeries, or LUW server”](#) on page 23 for the next step in the installation process.

## Jobs that migrate QMF V7.2 or earlier to QMF V12.1 in a VM or VSE server

This job sequence migrates QMF Version 7.2 or earlier to QMF Version 12.1 in a DB2 for VSE and VM server.

### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. This procedure is associated with Installation path G: Migrating to QMF V12.1 from V7.2 (or earlier) in a DB2 for VSE and VM server.
- Ensure that the DB2 for VSE and VM is Version 7.3 or later.

### Procedure

- Customize parameter values in the jobs as necessary by following the instructions in the prolog of each job or by setting site-specific installation defaults or by overriding installation defaults. Then run the jobs in their specified order.

The jobs listed in the following table reside as members in the QMF1210.SDSQSAPE data set.

Job name	Description
DSQ1BSQL	Binds the QMF installation programs from the Db2 for z/OS requester database to the server database  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID='secauth' specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.
DSQ1BVW	Creates the default views that support the QMF LIST command
DSQ1BPKG	Binds the QMF packages

- After you run each job, check for a return code of 0 or 4. Both codes indicate successful completion.
- If necessary, install QMF sample tables by following the documentation for QMF for VSE or VM. Previous releases of the QMF sample tables are still valid in the new QMF version.

### What to do next

After all jobs have run successfully, return to [“Installation path E: Migrating to QMF V12.1 from V7.2 or earlier in a DB2 for VSE and VM server”](#) on page 25 for the next step in the installation process.

### Related information

Search for information about installing the QMF sample tables in the QMF for VM or VSE documentation.

## Jobs that migrate QMF V8, V9, V10, V11.1 or V11.2 to QMF V12.1 in a VM or VSE server

This job sequence migrates an existing QMF Version 8.1, Version 9.1, Version 10, Version 11.1, or Version 11.2 installation to QMF Version 12.1 in a DB2 for VSE and VM server.

### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. This procedure is associated with Installation path H: Migrating to QMF V12.1 from QMF V8.1, V9.1, V10, V11.1, or V11.2 in a DB2 for VSE and VM server.



- Ensure that the DB2 for VSE and VM is Version 7.3 or later.

### About this task

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

### Procedure

- Customize parameter values in the jobs as necessary by following the instructions in the prolog of each job or by setting site-specific installation defaults or by overriding installation defaults. Then run the jobs in their specified order.

The jobs listed in the following table reside as members in the QMF1210.SDSQSAPE data set.

<i>Table 21. Job sequence for migrating an existing QMF Version 8.1, Version 9.1, Version 10, Version 11.1, or Version 11.2 installation to QMF Version 12.1 in a DB2 for VSE and VM server</i>	
Job name	Description
DSQ1BSQL	Binds the QMF installation programs from the Db2 for z/OS requester database to the server database.  If you are installing QMF under a secondary authorization ID, edit this job and add either the OWNER( <i>secauth</i> ) parameter or a SET CURRENT SQLID='secauth' specification to the BIND PACKAGE and BIND PLAN statements so that QMF is able to resolve the ID. The value of the SECAUTH parameter in the DSQ1DEFS exec must match the SECAUTH value that you provide in this job.
DSQ1BPKG	Binds the QMF packages.

- After you run each job, check for a return code of 0 or 4. Both codes indicate successful completion.
- If necessary, install QMF sample tables by following the documentation for QMF for VSE or VM. Previous releases of the QMF sample tables are still valid in the new QMF version.

### Related information

Search for information about installing the QMF sample tables in the QMF for VM or VSE documentation.

## Falling back to the prior release in server databases

If you encounter errors in migrating to QMF Version 12.1 from a QMF Compatibility Mode release (QMF Version 3.3, 6.1, 7.1, 7.2, Version 8.1 Compatibility Mode, or Version 9.1 Compatibility Mode), you can return to the earlier release.

### Procedure

To return the QMF control tables and their table spaces to their original pre-migration state:

1. Re-create the QMF control tables with their original structure. You can use either of the following methods:
  - Drop the migrated objects and re-create them with their original structure and data. Use your backup that you created when you migrated.
  - Use ALTER statements to change the definition of the objects to match the original specifications as they existed before the migration.
2. Recover the original image copies.

You must recover the original image copies because the migration process for QMF Version 12.1 renames the original control tables and drops the renamed tables. If you are migrating QMF on a Db2 for z/OS server, use the DSN1COPY utility rather than the RECOVER utility to recover the original image

copies. Because the OBIDs of the original image copies do not match the OBIDs of the newly created tables, the OBIDLAT parameter is required.

See the Db2 documentation for more information about the DSN1COPY utility.

**Note:** In DB2 Version 9 and later, simple table spaces are supported, but new tables are created only in segmented table spaces. If the QMF tables were created before DB2 Version 9, use the DB2 UNLOAD utility with the fix for APAR PK60612 applied instead of DSN1COPY. With this fix, the UNLOAD utility can process an image copy of a table space that was non-segmented although the new table is defined as segmented. For more information, see the "Unloading from an image copy" section of [DB2 9 for z/OS: Using the Utilities Suite](#).

3. Recover any indexes on the objects.
4. Take a full image copy of the re-created objects from the prior release.

## Running installation jobs that prepare servers to be accessed by QMF commands that include three-part names

---

This job sequence prepares both the requester and server to use three-part names in QMF commands to access data at a remote server.

### Before you begin

- Ensure that you are following the correct installation path before you perform these tasks. This procedure is associated with Installation path I: Preparing a remote server to be accessed by QMF commands that include three-part names.
- If you intend to access data at a remote server by using the QMF CONNECT command in addition to QMF commands that include three-part names, see [“Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command”](#) on page 19 to determine your installation path.

### About this task

#### Restrictions:

- Connectivity with remote servers is not supported when QMF for TSO is started as a stored procedure in the Db2 for z/OS requester database.
- QMF commands that include three-part names cannot be directed to DB2 for VSE and VM servers.
- By default, QMF commands that include three-part names cannot be used to access remote tables that contain large object (LOB) data. To enable access of LOB data in remote tables with three-part names, the DSQEC\_LOB\_RETRV global variable must be set to 3. The DSQEC\_LOB\_RETRV can also be set to 2 to enable the retrieval of LOB metadata only.
- Unless the command that includes the three-part name is directed to Db2 for z/OS, QMF must be started with multirow fetch turned off.

### Procedure

To use only three-part names in QMF commands to access data at a remote server where QMF is not installed, complete the following steps to prepare both the requester and server:

1. Run QMF1210.SDSQSAPE(DSQ1BSQL) at the remote server. Run this job from the local database.
2. Edit QMF1210.SDSQEXCE(DSQ1DEFS) and set the T3PARTNM parameter to YES.  
For example:

```
T3PARTNM="YES"
```

3. Be sure that the DSQDEFS DD statement in job DSQ1BPKG points to the DSQ1DEFS member that you modified in step “2” on page 68.
4. Run QMF1210.SDSQSAPE(DSQ1BPKG) at the remote server. Run this job from the local database. Check for a return code of 0 or 4. Both codes indicate successful execution.

**Related concepts**

Enabling support for multirow fetch and insert

The DSQSMRFI parameter controls whether the database uses multirow or single-row fetch and insert.

**Related tasks**

Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.



---

# Chapter 5. Verifying that QMF is installed correctly

The methods to verify that QMF is correctly installed vary depending on the type of database.

## Testing QMF installations on requester databases

---

To test installations of QMF in requester databases, complete these steps.

### Before you begin

- Ensure that you have completed all of the steps in the installation procedure that is associated with your chosen installation path.
- Ensure that you have the proper authority to run the tests.

The authorization ID that you use to run the tests must have one of the following authorities:

- SYSADM (or equivalent)
- DBADM (or equivalent) on the following databases:

#### **DSQDBCTL**

This database contains the QMF control tables of the QMF object catalog.

#### **DSQDBDEF**

Depending on your installation, this database may or may not contain the DSQTSDEF table space. If DSQTSDEF table space is present, it is used as the default space when the SAVE DATA command is issued. If DSQTSDEF table space is not present, SAVE DATA will create tables in implicitly created universal table spaces.

### About this task

These test procedures apply to the installation paths in [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)”](#) on page 14.

If you plan to use QMF under CICS, run all steps under TSO first, then go through the procedure again for CICS.

## Starting QMF

The method that you use to start QMF depends on whether you are running QMF under TSO or CICS.

### Before you begin

Before starting QMF, verify that the requester database has been started. Start the requester database with DDF if you will proceed with a QMF installation in a server database after testing the requester installation.

### About this task

Separate instructions are provided for starting QMF under TSO and starting QMF under CICS.

### Results

If QMF fails to start or you experience errors or warnings when starting QMF, see one of the following topics for possible causes:

- [“Errors that can occur at initialization time”](#) on page 336
- [“Warning messages after you start QMF”](#) on page 344

Initialization problems can occur if QMF is accessing load modules from one or more prior releases during startup. See [“Problems that can occur when QMF is not using current load modules”](#) on page 336 for instructions on how to investigate whether this might be the problem.

### Starting QMF under TSO

If you are using QMF under ISPF, you can start QMF by using the ISPF SELECT service and the ISPSTART command. To start QMF without ISPF, you can use the DSQQMF $n$  module, where  $n$  is a 1-character language ID.

### About this task

After logging onto TSO with a logon procedure, you are in TSO READY mode. From this mode, you can start QMF with or without ISPF.

### Related concepts

[Setting program parameters and preferences at startup time](#)

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

### Related tasks

[Starting QMF under CICS](#)

As part of verifying that QMF is installed correctly under CICS, ensure that QMF starts correctly

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

[Preparing the TSO logon procedure](#)

You must modify the TSO logon procedure to support the storage and other requirements of your users. This step includes allocating load libraries and data sets.

### Related information

Search for information about ISPF dialogs in the ISPF information.

### Starting QMF with ISPF

You can start QMF by using the ISPSTART command or you can customize ISPF selection menus to include QMF as one of the menu choices.

### Procedure

- Start QMF with ISPF by either of these methods:
  - To use the ISPSTART command, start QMF from an application program by using the callable interface, or issue the ISPSTART command with or without parameters.

The following example shows how to use the ISPSTART command to override the default values for the database subsystem name (DSN) and the plan ID (QMF12).

```
ISPSTART PGM(DSQQMF $n$ ) NEWAPPL(DSQ $n$ ) PARM(DSQSSUBS= $ssid$ ,  
DSQSPLAN= $planid$ ,...)
```

This command includes the following variables:

**$n$**

Use the appropriate 1-character language ID to start QMF in your NLF environment. For example, to start QMF in an English environment, the values of the PGM parameter is DSQQMFE and the value of the NEWAPPL parameter is DSQE.

**$ssid$**

The 4-character name of the Db2 for z/OS subsystem where QMF Version 12.1 is installed.

**$planid$**

The name of the QMF application plan that is bound to the requester. The default is QMF12.

After you issue the preceding ISPSTART command, the QMF home panel is displayed, as shown in “Checking for the correct version of the QMF panel library” on page 75.

- To code the ISPF Master Application menu supplied with ISPF to include QMF as one of the ISPF selection menus choices, follow this example:

The call that is highlighted in the lower half of the figure passes a Db2 for z/OS subsystem name of DB2P and a plan name of QRY10.

```
%----- MASTER APPLICATION MENU -----
%SELECT APPLICATION ==>_OPT      +
%                               +USERID  -
%                               +TIME    -
%    1 +SPF                      - SPF PROGRAM DEVELOPMENT FACILITY    +TERMINAL -
%    2 +QMF                      - QMF QUERY MANAGEMENT FACILITY      +PF KEYS  -
%
%
%
%
%
%
%
%
% P +PARMS      - SPECIFY TERMINAL PARAMETERS AND LIST/LOG DEFAULTS
% X +EXIT       - TERMINATE USING LIST/LOG DEFAULTS
%
+PRESS%END KEY+TO TERMINATE +
%
)INIT
)PROC
  &SEL = TRANS( TRUNC (&OPT,'.')
                1,'PANEL(ISR@PRIM) NEWAPPL '
                2,'PGM(DSQMF) NEWAPPL(DSQE) PARM(S=DB2P,P=QRY10) '
                /* ADD OTHER APPLICATIONS HERE */
                P,'PANEL(ISPOPT) '
                X,'EXIT'
                ',',' '
                *,'? ' )
)END
```

Figure 13. QMF dialog on the ISPF Master Application menu (English shown)

If you are starting QMF in an NLF environment, you can add entries to the menu for the NLF. In the example, replace the "E" in DSQMF and DSQE with the 1-character national language identifier for your NLF.

You can also customize any other ISPF menu to allow users to invoke QMF by selecting it from the menu.

## Related concepts

### [Setting program parameters and preferences at startup time](#)

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

## Related tasks

### [Starting QMF without ISPF](#)

To start QMF without ISPF, you can use an application program and the QMF callable interface, or you can use TSO CALL commands.

### [Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### **Starting QMF without ISPF**

To start QMF without ISPF, you can use an application program and the QMF callable interface, or you can use TSO CALL commands.

### **Procedure**

- To use an application program and the callable interface, enter the following statement: `DSQQMFn DSQSSUBS=ssid,DSQSPLAN=planid, ...`

This command includes the following variables:

#### **n**

Use the appropriate 1-character language ID to start QMF in your NLF environment. For example, to start QMF in an English environment, the command is DSQQMFE.

#### **ssid**

The 4-character name of the Db2 for z/OS subsystem where QMF Version 12.1 is installed.

#### **planid**

The name of the QMF application plan that is bound to the requester. The QMF Version 12.1 application plan ID is QMF12.

You can also specify additional values in place of the ellipsis shown.

- To use TSO CALL commands, you can use a TSO CALL command to initialize QMF in the local Db2 for z/OS requester database.

For example:

```
CALL 'QMF1210.SDSQLOAD(DSQQMFn)' 'DSQSSUBS=ssid,DSQSPLAN=planid,...'
```

This command includes the following variables:

#### **QMF1210**

The default prefix for the QMF Version 12 Release 1 libraries. If you changed the default prefix for the QMF libraries, substitute that prefix in place of QMF1210.

#### **n**

Use the appropriate 1-character language ID to start QMF in your NLF environment. For example, to start QMF in an English environment, use DSQQMFE.

#### **ssid**

The 4-character name of the Db2 for z/OS subsystem where QMF Version 12.1 is installed.

#### **planid**

The name of the QMF application plan that is bound to the requester. The default is QMF12.

### **Related concepts**

[Setting program parameters and preferences at startup time](#)

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

### **Related tasks**

[Starting QMF with ISPF](#)

You can start QMF by using the ISPSTART command or you can customize ISPF selection menus to include QMF as one of the menu choices.

[Installing QMF National Language Features](#)



A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### Starting QMF under CICS

As part of verifying that QMF is installed correctly under CICS, ensure that QMF starts correctly

#### Procedure

To start QMF for CICS, follow these steps:

1. Sign onto the CICS system where QMF is installed.
2. Press the Escape key to begin a native CICS session.
3. Start QMF:
  - a) Issue the CICS transaction QMF *n*, replacing *n* with the 1-character national language identifier for your NLF.  
For example, for English, the transaction name is QMFE.
  - a) Specify program parameters DSQSDBQT and DSQSDBQN to configure the type and name of the queue that will hold trace data.  
For example, to start QMF with a temporary storage queue name of DSQD, issue the following command:

```
QMFn DSQSDBQT=TS,DSQSDBQN=DSQD
```

#### Related tasks

##### Starting QMF under TSO

If you are using QMF under ISPF, you can start QMF by using the ISPF SELECT service and the ISPSTART command. To start QMF without ISPF, you can use the DSQQMF*n* module, where *n* is a 1-character language ID.

##### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Checking for the correct version of the QMF panel library

Both the home panel and help panels should display correctly.

#### Procedure

To check that the correct version of the QMF panel library is being accessed, complete the following steps:

1. Check that the correct version of the QMF home panel is displayed, as shown in the following figure. On the home panel:
  - a) Check that "Version 12 Release 1" displays beneath the panel name **(A)**.  
If the version and release information is not correct and you started QMF under TSO, check your allocation for DD statement ADMGGMAP to make sure that it points to QMF1210.SDSQMAP*n*. In place of specify the 1-character national language ID that is associated with your NLF.  
For example, for English QMF, the DD statement should be pointing to QMF1210.SDSQMAPE.  
If the version and release information is not correct and you started QMF under CICS, check that the ADMF DD statement in the startup JCL for the CICS region in which QMF resides correctly points to the GDDM ADMF data set. The DSQ1*n*ADM installation job (DSQ1EADM for English QMF) writes the QMF Version 12.1 GDDM maps to this data set.
  - b) Check that you are connected to the Db2 for z/OS database into which you just installed QMF **(B)**. The following example shows that the database is named DB2P.

```

-----
-----
QMF HOME PANEL                               Query      Management  Facility
Version 12 Release 1 A

Authorization ID                               ***** ** ** *****
KMAMEY                                         ** ** *** *** **
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
Connected to                                   ** * ** ** ** ** ** ** ** ** **
DB2P B                                         ***** ** ** ** **
**
http://www.ibm.com/qmf
Enter a command on the command line or press a function key.
For help, press the Help function key or enter the command HELP.
-----
1=Help      2=List      3=End      4=Show      5=Chart      6=Query
7=Retrieve  8=Edit Table 9=Form     10=Proc     11=Profile   12=Report
OK, you may enter a command.
COMMAND ==>

```

Figure 14. QMF Version 12.1 home panel

- 2. Verify the existence of Version 12.1 online help.
  - a) Press the Help function key.

The main help menu for QMF Version 12.1 is displayed, as shown in the following figure. Be sure that topic 1 reflects the correct version of QMF.

```

+-----+
|           Help: Query Management Facility           |
| Select a topic.                                     |
|                                                     |
|           1 to 11 of 17                             |
| 1. What's new in QMF Version 12 Release 1         |
| 2. Learning about QMF                             |
| 3. Profile                                         |
| 4. QMF commands                                   |
| 5. Prompted Query                                 |
| 6. SQL (Structured Query Language)                 |
| 7. Table Editor                                   |
| 8. Forms                                          |
| 9. Reports                                        |
| 10. Charts                                       |
| 11. Data analytics                               |
|-----|
| F1=Help F3=Exit F7=Backward F8=Forward           |
| F9=Keys F12=Cancel                               |
|-----+

```

Figure 15. Main help menu in QMF Version 12.1

- b) Press Cancel (F12 by default) to return to the home panel.

**What to do next**

If the version and release information is incorrect or the help does not display at all, verify that the DSQ1nPNL installation job ran successfully and that the DD statement for the DSQPNLn file points to the correct Version 12 Release 1 data set. In place of n, specify the 1-character national language ID that is associated with your NLF.

If you started QMF under CICS, additionally check the CICS startup job stream to ensure that the DSQPNLn data set is properly allocated.

**Related concepts**

[Populating the VSAM panel library](#)

When the SMP/E installation is complete, populate the QMF VSAM panel library, DSQPNLE.

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Checking that QMF libraries have been allocated correctly

To check that QMF Version 12.1 libraries have been properly allocated, issue the SHOW GLOBALS command from within QMF.

Scroll through the global variables and check the value of the following variables:

- DSQAO\_QMF\_VER\_RLS

The value of this variable should be QMFV12R1.0. If any other value is displayed, review the information on allocating QMF libraries in [“Defining programs, resources, and preferences to TSO and CICS”](#) on page 49.

### Related concepts

#### Initializing global variables and QMF session behavior when QMF starts

You can use several methods to set QMF global variables when QMF starts. These methods involve modifying the DSQUOPTS routine, the Q.GLOBAL\_VARS global variable table, or the default system initialization procedure. This procedure can also be used to set other aspects of a user's QMF session before the home panel displays.

## Checking that all required QMF views were installed

Check that the views are appropriate for the database to which you are connected.

### Procedure

To verify that the appropriate views were installed:

1. Issue the following command:

```
LIST TABLES (OWNER=Q NAME=DSQ%)
```

2. Check the results of the command against the views that are shipped with QMF.
3. If the appropriate views are not in the list, rerun the DSQ1BVW installation job. If you installed the enhanced LIST command function, run the DSQ1BUDV installation job after you run DSQ1BVW.

### Related reference

#### QMF views

These views are shipped with QMF. QMF uses these views on the platforms indicated in the table to create object lists when the QMF LIST command is issued.

## Installing sample queries and procedures required for the installation verification procedures

Install the sample QMF queries and procedures only on TSO.

Some of these objects are required by the installation verification procedures for both TSO and CICS. After the objects are installed under TSO, they are then available for use under CICS. Other objects are provided for users to refer to while learning how to use QMF.

The sample queries and procedures have changed in QMF Version 12.1, so you must delete prior versions and install the new versions.

The following table describes the sample queries.

Table 22. QMF Version 12.1 sample queries

<b>Name of data set from which query is imported</b>	<b>Name under which QMF saves the object after importing it</b>	<b>Description</b>
DSQ0EQ1	Q.SAMPLE_SELECT1	Simple SELECT query
DSQ0EQ2	Q.SAMPLE_SELECT2	SELECT query with IN keyword and ORDER BY clause
DSQ0EQ3	Q.SAMPLE_SELECT3	SELECT query with BETWEEN and NULL keywords and AND and OR operators
DSQ0EQ4	Q.SAMPLE_SELECT4	SELECT query that uses LIKE keyword
DSQ0EQ5	Q.SAMPLE_SELECT5	SELECT query that does a join of two tables
DSQ0EQ6	Q.SAMPLE_SELECT6	SELECT query that uses MAX, MIN, and AVG functions
DSQ0EQ7	Q.SAMPLE_SELECT7	SELECT query with GROUP BY and HAVING clauses; COUNT function
DSQ0EQ8	Q.SAMPLE_SELECT8	SELECT query that uses a UNION operator
DSQ0EQ9	Q.SAMPLE_SELECT9	SELECT query that uses a subquery
DSQ0EQ10	Q.SAMPLE_SELECT10	SELECT query with correlated subselection
DSQ0EQ11	Q.SAMPLE_SELECT11	SELECT query with EXISTS keyword
DSQ0EQ12	Q.SAMPLE_INSERT1	INSERT query to add a single row to a table
DSQ0EQ13	Q.SAMPLE_INSERT2	INSERT query to copy one or more rows into a table
DSQ0EQ14	Q.SAMPLE_UPDATE	UPDATE query
DSQ0EQ15	Q.SAMPLE_DELETE	DELETE query
DSQ1EQ16	Q.SAMPLE_DATABASE	Query that uses the CREATE DATABASE statement
DSQ1EQ17	Q.SAMPLE_TABLESPACE	Query that creates a table space
DSQ1EQ18	Q.SAMPLE_CREATETABLE	Query that creates a table
DSQ0EQ19	Q.SAMPLE_CREATEINDEX	CREATE INDEX query
DSQ0EQ20	Q.SAMPLE_CREATEVIEW	Query that creates a view
DSQ0EQ21	Q.SAMPLE_GRANT	Query that grants update privileges
DSQ0EQ22	Q.SAMPLE_REVOKE	Query that revokes update privileges
DSQ0EQ23	Q.SAMPLE_DROP	Query that drops a table from a database
DSQ0EQ25	Q.SAMPLE_COMMENT	Query that saves a comment with a table

The following table describes the sample procedures.

Table 23. QMF Version 12.1 sample procedures

Name of data set from which procedure is imported	Name under which QMF saves the object after importing it	Description
DSQ0BIQ1	Q.IVP_QUERY_COUNT	Provides a row count for a given input table.
DSQ0BIQ2	Q.IVP_QUERY_DELETE	Deletes QMF_IVP_QUERY from the catalog
DSQ1EIVP	Q.DSQ1EIVP	IVP procedure to verify interactive operation under TSO
DSQ1EBAT	Q.DSQ1EBAT	Batch IVP procedure to verify batch operation under TSO
DSQ1EIVC	Q.DSQ1EIVC	CICS IVP procedure
DSQAED1S	Q.DSQAED1S	Installs the document editing interface function
DSQABL01 DSQAEL0D DSQABL0D	Q.DSQABL01 Q.DSQAEL0D Q.DSQABL0D	Installs procedures for the LAYOUT application, which supports the LAYOUT command synonym

Complete the following steps to delete sample queries and procedures from the prior QMF version and install the QMF Version 12.1 objects:

1. Begin a QMF session.
2. Make sure that you are connected to the Db2 for z/OS subsystem in which you just installed QMF.
3. Delete prior versions of the sample queries and procedures. If you installed QMF Version 12.1 in a system where no previous QMF release existed, skip this step.
  - a. Enter the following command to display the procedure that deletes the queries and procedures:

```
IMPORT PROC FROM QMF1210.SDSQSAPE(DSQ1ESQD) '
```

In this command and throughout the DSQ1ESQD procedure, QMF1210 is the prefix for the QMF Version 12.1 data sets. If you used a different prefix, change QMF1210 in the command and wherever it occurs in the procedure to the site-specific prefix that you have chosen. If you are using QMF on TSO, you can use the EDIT PROC command, followed by an ISPF command such as CHANGE, to change the prefix information. For example, the following ISPF command changes the prefix from QMF1210 to the prefix DB2TOOLS:

```
CHANGE 'QMF1210' 'DB2TOOLS' ALL
```

- b. Issue the RUN PROC command to run the procedure.
4. Install the Version 12.1 sample queries and procedures.

Enter the following command to display the procedure that installs the Version 12.1 queries and procedures:

```
IMPORT PROC FROM 'QMF1210.SDSQSAPE(DSQ1ESQI) '
```

In this command and throughout the DSQ1ESQI procedure, QMF1210 is the prefix for the QMF Version 12.1 data sets. If you used a different prefix, change QMF1210 in the command and wherever it occurs in the procedure to the site-specific prefix that you have chosen. If you are using QMF on TSO, you can use the EDIT PROC command, followed by an ISPF command such as CHANGE, to change the prefix information. For example, the following ISPF command changes the prefix from QMF1210 to QMF12:

```
CHANGE 'QMF1210' 'QMF12' ALL
```

5. Issue the RUN PROC command to run the procedure.

You should receive a message that indicates that the objects were installed correctly. If a failure occurs, you can use the DSQ1ESQD procedure to delete any already created objects and begin again.

Message DSQ21662 is an informational message only and does not indicate errors; you can ignore this message and proceed to run the IVPs.

## Running the installation verification procedures

The installation verification procedure (IVP) for QMF installations on requester databases differs slightly depending on whether the installation in the requester database is running under TSO or CICS.

### About this task

To test that QMF Version 12.1 is installed correctly, you must run the IVPs that you installed in [“Installing sample queries and procedures required for the installation verification procedures” on page 77](#).

### Running the IVPs for TSO

There are two installation verification procedures for QMF installations that run under TSO: one that tests interactive operations and one that tests batch operations.

### About this task

If you plan to use QMF in batch mode, complete both test procedures.

### Related tasks

[Running the IVP for CICS](#)

To test for proper operation of QMF under CICS, complete these steps.

### Testing interactive operation under TSO

To test interactive operation of QMF under TSO, complete these steps.

### Before you begin

If you have not done so already, start QMF.

### Procedure

1. Display the QMF installation verification procedure for TSO by issuing the following command:  
`DISPLAY PROC Q.DSQ1EIVP.`

This procedure was imported and saved in the database in the process explained in [“Installing sample queries and procedures required for the installation verification procedures” on page 77](#). Comments at the beginning of the procedure describe what the procedure tests.

2. On the **PROC** panel, customize the procedure if necessary.

QMF1210 is the prefix for the QMF Version 12.1 data sets. If you used a different prefix, change QMF1210 throughout DSQ1EIVP to the site-specific prefix that you have chosen.

If you change the procedure, issue the following command to save it: `SAVE PROC AS Q.DSQ1EIVP (SHARE=YES.`

3. Run the procedure by issuing the following command: `RUN PROC.`

Enter a value of 1 in response to each prompt.

### Results

When the IVP completes successfully, the following message is displayed: The QMF IVP procedure has successfully completed.

Exit the QMF session by issuing the EXIT command.

## What to do next

If the IVP does not complete successfully, review the output from the QMF installation jobs that you ran to ensure that all jobs completed successfully. You can also use the QMF messages and message help panels (which are displayed when you press the Help key) to diagnose and correct the problem.

### Testing batch operation under TSO

To test batch operation of QMF under TSO, complete these steps. Skip this step if you do not plan to run QMF in batch mode.

### About this task

A sample procedure named DSQ1*n*BAT (where *n* is a 1-character national language identifier) is shipped with QMF to test batch operation. The procedure verifies that the QMF control tables exist and that basic QMF operations complete successfully.

You can use other methods to start QMF in batch mode and run the Q.DSQ1*n*BAT procedure. If the following method does not suit your needs, see [Chapter 9, “Starting QMF,” on page 117](#) for additional options.

### Procedure

Complete the following steps to test batch operation of QMF under TSO by using the DSQ1*n*BAT procedure:

1. If you have not done so already, start QMF.
2. When the QMF home panel is displayed, display the Q.DSQ1*n*BAT procedure.  
For example, to display the English batch IVP, issue the following command:

```
DISPLAY Q.DSQ1EBAT
```

If the procedure cannot be found, see [“Installing sample queries and procedures required for the installation verification procedures” on page 77](#) for steps on installing it. If you are testing batch mode for a QMF NLF, see Step [“10” on page 105](#) of the procedure in [“Installing QMF National Language Features” on page 101](#) for information about how to install the batch-mode IVP.

3. Review the procedure and customize it if necessary.

QMF1210 is the prefix for the QMF Version 12.1 data sets. If you used a different prefix, change QMF1210 throughout the procedure to the site-specific prefix that you have chosen.

If you change the procedure, issue a command like the following to save it:

```
SAVE PROC AS Q.DSQ1EBAT (SHARE=YES
```

4. Customize the TSO logon procedure to start QMF in batch mode and run the Q.DSQ1*n*BAT procedure:
  - a) Make a copy of the sample logon procedure (DSQ1EINV).
  - b) Add a JOB statement that is appropriate for your site's needs.
  - c) Delete the SYSTEM and SYSIN DD statements.
  - d) Set up printing capabilities.

The QMF procedure that you will be running in batch mode tests the QMF PRINT command. To ensure that this command is successful, add DD statements that allocate the DSQPRINT file. Output from the QMF PRINT command goes to this file. For example:

```
//DSQPRINT DD SYSOUT=A
```

- e) Specify how to control the output from batch mode by adding a statement similar to the following to the end of the logon procedure:

```
//SYSTSPRT DD SYSOUT=A
```

- f) Add the following statements after the SYSTSPRT statement.

Replace *n* with the 1-character national language identifier that corresponds to the language that you are testing.

```
//SYSTSIN DD *  
    PROFILE PREFIX(username)  
    ISPSTART PGM(DSQMFn) NEWAPPL(DSQn) PARM(M=B,I=Q.DSQ1nBAT,S=ssid)  
/*
```

The first control card within the second JCL statement is required only if your installation does not use RACF. Replace *username* with the logon ID of the user who is running the step. See [“JCL to execute a QMF batch job” on page 320](#) for more information about this statement.

The second control card within the second JCL statement invokes QMF in batch mode (M=B). Replace *ssid* with the 4-character subsystem ID of the Db2 for z/OS subsystem into which you installed QMF. If you do not specify a subsystem ID, the default, DSN, is used. When invoked in this way, QMF runs the Q.DSQ1*n*BAT procedure.

For more information about other QMF program parameters that you can pass on the ISPSTART command, see [Chapter 10, “Setting program parameters and preferences at startup time,” on page 141](#).

5. Save the revised TSO logon procedure.
6. Submit the job, which starts QMF and submits the Q.DSQ1*n*BAT procedure as a batch job.

After the procedure runs, control returns to TSO, which terminates the job because it finds no more TSO statements in SYSTSIN. Examine the job output to ensure that you received no errors. The following messages are displayed in the trace output when the job completes successfully:

```
OK, TSO command executed successfully.  
OK, your procedure was run.
```

If the IVP does not complete successfully, review the output from the QMF installation jobs that you ran to ensure that all jobs completed successfully. You can also use the QMF messages and message help panels (which are displayed when you press the Help key) to diagnose and correct the problem.

## Related concepts

### [Running QMF in batch mode](#)

If a user runs a procedure with the RUN command, the user cannot execute QMF commands except to cancel the procedure or the session. Thus, running a procedure using the RUN command can tie up considerable session time.

## Related tasks

### [Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Running the IVP for CICS

To test for proper operation of QMF under CICS, complete these steps.

## Procedure

1. Verify that the QMF trace facility is installed by checking the default transient data queue that holds the trace data (DSQD). From a clear CICS screen, enter the following command: CEMT INQUIRE QUEUE(DSQD).

Output similar to the following is displayed:

```
STATUS:      RESULTS  - OVERTYPE to MODIFY  
Que(DSQD)   Ext Ena Ope
```



Ena Ope indicates that the queue is open and enabled. If you do not see that DSQD is enabled and open, review your changes to the CICS system definition file. See [“Describing QMF to CICS”](#) on page 56 for details.

2. If you have not done so already, start QMF as explained in [“Starting QMF”](#) on page 71.
3. Display the QMF installation verification procedure for CICS by entering the following command:  
`DISPLAY PROC Q.DSQ1EIVC.`  
This procedure was imported and saved in the database in the process explained in [“Installing sample queries and procedures required for the installation verification procedures”](#) on page 77. Comments at the beginning of the procedure describe what the procedure tests.
4. On the **PROC** panel, customize the procedure if necessary. If you change the procedure, issue the following command to save it: `SAVE PROC AS Q.DSQ1EIVC (SHARE=YES).`
5. Run the procedure by entering the following command: `RUN PROC.`  
Enter a value of 1 in response to each prompt.

## Results

When the IVP completes successfully, the following message is displayed: The QMF IVP procedure has successfully completed.

Exit the QMF session by issuing the EXIT command.

If the procedure does not run successfully, review the output from the QMF installation jobs that you ran to ensure that all jobs completed successfully. To browse the temporary storage queue that you verified in Step [“1”](#) on page 82 for any QMF warning messages, issue the following CICS transaction: CEBR DSQD

## Related tasks

### [Running the IVPs for TSO](#)

There are two installation verification procedures for QMF installations that run under TSO: one that tests interactive operations and one that tests batch operations.

## Testing QMF installations on server databases

---

You test installations of QMF on server databases as part of the installation paths for server databases accessed by the QMF CONNECT command or the DSQSDBNM program parameter.

### Before you begin

Before you test an installation of QMF on a server database, ensure that you have tested the installations on requester databases:

- Test the installations of QMF on all requester databases that will access the server.
- Also complete the test procedure for requesters if the remote server database is Db2 for z/OS. For information about the procedure, see [“Testing QMF installations on requester databases”](#) on page 71

### About this task

These test procedures apply to the installation paths in [“Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command”](#) on page 19.

### Procedure

To test a QMF installation on a server database other than z/OS, complete the following steps:

1. Start QMF in the Db2 for z/OS requester database with distributed data facility (DDF).
  - If QMF is running in TSO, see [“Starting QMF under TSO”](#) on page 72.
  - If QMF is running in CICS, see [“Starting QMF under CICS”](#) on page 75.

The DSQSSUBS program parameter specifies the name of the Db2 for z/OS requester on which you are starting QMF.

2. Connect to the server database. You can connect to QMF in the server database in one of two ways:
  - After you start QMF in the requester database and the QMF home panel is displayed, use the QMF CONNECT command to connect to the server.
  - Specify the DSQSDBNM program parameter when you start QMF in the requester database. QMF connects to the server location specified on this parameter before displaying the home panel. For more information about this parameter, see [“Specifying the initial database connection”](#) on page 149.

For example, to start QMF in a local Db2 for z/OS database called LOCL and immediately connect to a remote Db2 for Linux, UNIX, and Windows database called REMT, specify the following program parameters on the command that you use to start QMF:

```
DSQSSUBS=LOCL , DSQSDBNM=REMT
```

3. Check that the database name shown beneath Connected to on the QMF home panel is the correct name of the remote database server.  
For an example of the home panel, see [Figure 14 on page 76](#).
4. Check that QMF views have been correctly installed. Issue this command to check that the views that you see in the list are appropriate for the type of database to which you are connected: LIST TABLES (OWNER=Q NAME=DSQ%.  
[“QMF views”](#) on page 388 lists, by platform, the default views that support the QMF LIST command.  
If the appropriate views are not in the list, rerun the DSQ1BVW installation job. If the server is Db2 for z/OS and you installed the enhanced LIST command function, run the DSQ1BUDV installation job after you run DSQ1BVW.
5. Return to the QMF home panel by pressing the Cancel function key (F12 by default). End the QMF session by pressing the End function key (F3 by default).

### What to do next

After completing these tasks, return to the installation procedure that is associated with your chosen installation path to determine the next step in the installation.

### Related concepts

[Setting program parameters and preferences at startup time](#)

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

[Starting QMF](#)

QMF can be started only from z/OS. QMF can be set up to run under TSO, ISPF, as a batch job, or under CICS.

### Related tasks

[Testing QMF installations on requester databases](#)

To test installations of QMF in requester databases, complete these steps.

---

## Chapter 6. Migration and fallback considerations

After you install or migrate to QMF Version 12.1, you might need to access and use objects from earlier releases of QMF to thoroughly test the new installation under site-specific conditions.

You also might need to use objects or programs that you created under QMF Version 12.1 with earlier releases of QMF if you decide to fall back to an earlier release for any reason.

### Related tasks

[Falling back to the prior release in server databases](#)

If you encounter errors in migrating to QMF Version 12.1 from a QMF Compatibility Mode release (QMF Version 3.3, 6.1, 7.1, 7.2, Version 8.1 Compatibility Mode, or Version 9.1 Compatibility Mode), you can return to the earlier release.

---

### Accessing profiles and objects from a prior QMF release

How easily you can access objects created under an earlier release of QMF depends on whether the two releases reside in the same database.

If you installed QMF into a database that contains an earlier release of QMF, objects and programs from the earlier release are available under the new release.

If you installed QMF Version 12.1 in a database that does not contain an earlier release, but you need to access the objects from an earlier release that resides in a different database, you can re-create the profiles and migrate the tables, views, and QMF objects.

### Accessing QMF profiles in a different database

To use profiles from an earlier release that does not reside in the same database as the newly installed release, re-create the profiles.

#### About this task

To access QMF, users need a QMF profile. Profile information is stored in the Q.PROFILES control table.

#### Procedure

To use profiles from an earlier release that does not reside in the same database as the newly installed release, re-create the profiles in one of the following ways:

- If the earlier release is QMF Version 8.1 or Version 9.1 New Function Mode or QMF Version 10, Version 11.1 or Version 11.2, you can use the QMF EXPORT and IMPORT commands to migrate the profiles by exporting a copy of the Q.PROFILES table from the database that contains the earlier QMF release and importing it into the database that contains the new release of QMF.
- For any release of the prior installation of QMF, you can issue multiple INSERT statements from the **SQL Query** panel to re-create the appropriate data in the Q.PROFILES table.

#### Related concepts

[Creating QMF user profiles](#)

All QMF users need access to a user profile, which determines how QMF handles individual input from specific users. Use the profile to control certain aspects of a user's environment, such as where printer output is routed or whether input is converted to upper case.

#### Related tasks

[Migrating tables, views, and QMF objects](#)

You can use the QMF EXPORT and IMPORT commands to migrate tables, views, and QMF objects. Alternatively, you can use Db2 utilities.

## Migrating tables, views, and QMF objects

You can use the QMF EXPORT and IMPORT commands to migrate tables, views, and QMF objects. Alternatively, you can use Db2 utilities.

### Procedure

- To use QMF to migrate tables, views, and QMF objects, export the tables from the earlier release of QMF and then import them under the new release.

If you have Db2 QMF High Performance Option (HPO) installed and the database type is Db2 for z/OS, you can also use the HPO Object Manager to migrate objects from one Db2 subsystem to another.

- To use Db2 to migrate tables, views, and QMF objects, complete the following tasks
  - a) Unload the tables by using the tool or utility of your choice, such as the application program DSNTIAUL, supplied with Db2.
  - b) Load the tables into the Db2 subsystem where the new release of QMF is installed. Use a tool or utility of your choice, such as the DB2LOAD utility.
  - c) Use the available networking facilities to send the exported objects and unloaded tables to the system that contains the new release of QMF.

If the objects in question are not on Db2 for z/OS, use the system-specific utilities that are available to you for the type of database that you are working with.

### What to do next

Re-create any views, indexes, synonyms, and authorizations on the imported tables at the new database.

## Object compatibility between releases

---

Whether you can use objects created under a different release of QMF depends on whether the commands and functions that are referenced by the objects operate in the same way in both releases.

The installation process of QMF Version 12.1 supports migrations from Version 3.3, 6.1, 7.1, and 7.2, 8.1, 9.1, and 10.1. To help plan for any incompatibilities, make sure you understand the features and functions that were added or changed between the release from which you are migrating and QMF Version 12.1.

### Related concepts

[Summary of changes in prior releases](#)

QMF Version 12 Release 1 supports migrations from Version 8.1 NFM, Version 9.1 NFM, Version 10.1, Version 11.1 and Version 11.2. Enhancements in prior releases might affect how you plan your QMF migration.

## Forward compatibility of earlier releases with QMF Version 12 Release 1

Most objects that were created under earlier releases can be used in QMF Version 12.1 without modification.

However, you should be aware of some exceptions and considerations for using objects from earlier releases.

### QMF commands

- All QMF commands from prior releases can be issued in QMF Version 12.1 provided that the abbreviation and options are still valid in Version 12.1.

- For a list of commands, syntax, abbreviations, and options in Version 12.1, see the *Db2 QMF Reference* guide which is designed to help users, programmers, and database administrators understand this information:
  - The syntax and usage of the commands in QMF for TSO and CICS
  - How to use SQL keywords in queries
  - How to use forms, reports, and charts (including usage and edit codes)
- For more information about the QMF command interface, see the *Developing Db2 QMF applications* guide which includes information to help application programmers with:
  - Making application programming design decisions
  - Choosing between different programming techniques
  - Understanding how to use the QMF command and callable interfaces
  - Writing bilingual applications
- For more information about the document editing interface, see the *Using Db2 QMF* guide which is designed to help end users with:
  - Retrieving data
  - Formatting data into reports or charts
  - Editing data and other basic QMF tasks

### **QMF procedures**

QMF commands in procedures are subject to the restrictions that are explained in the QMF commands section.

Some procedures from earlier releases will not work properly if they issue commands with verbs that are also verbs for user-defined commands (called QMF command synonyms). To avoid potential problems, you can type QMF before all native QMF commands. For more information about command synonyms, see [“Customizing Command synonyms” on page 234](#).

### **Applications**

QMF commands in applications are subject to the restrictions that are explained in the QMF commands section.

## **Compatibility of QMF Version 12.1 with earlier releases**

Some objects that were created under QMF Version 12.1 cannot be used in earlier releases. Consider the incompatibilities when you plan for a possible fallback.

Unless otherwise stated, any compatibility issues with an earlier version apply to using objects in QMF Version 12.1 only. You can use the information in the summary of features to determine if functions that you use in Version 12.1 operate in the same way in a release earlier than Version 12.1.

For information about compatibility with an earlier version, see one of the following sections:

### **QMF commands**

The following commands or command options were added or changed in QMF Version 12.1 and therefore cannot be used in earlier releases. Procedures or applications that contain these commands or options must be modified before you attempt to run them under the earlier release.

- TABLE keyword on RUN QUERY for QBE and prompted queries

The following global variables are new in QMF Version 12.1. Commands that reference these variables cannot be used in earlier releases.

- DSQEC\_DS\_SUPPORT: This global variable allows QMF access to QMF Data Service data sources.
- DSQEC\_BUFFER\_SIZE: This global variable changes the QMF internal storage area buffer size.

- DSQEC\_TRACE\_MODULE: This global variable allows administrators to limit trace diagnostics by QMF module name.

### SQL statements

Queries referencing three part names to QMF Data Service data sources cannot be used in earlier releases. Modify any queries, procedures, or applications that include this statement before you fall back to an earlier release.

### Forms

Forms specifying the 'C' or 'CW' edit code for binary column data might not be usable in earlier QMF releases.

### QMF program parameters

Earlier QMF releases may not be started with the DSQSMTHD program parameter. Any queries, procedures, or applications depending on multiple thread support added in Version 12.1 might not be usable in earlier releases.

## Above the bar storage requirement changes

---

Above the bar storage is required for LOB and XML data type retrieval. For the LOB and XML data type and the 64 bit spill support, you can increase the amount of above the bar storage (MEMLIMIT).

QMF utilizes above the bar storage (storage above the 2-gigabyte bar). To control the amount of real and auxiliary storage that an address space can use for memory objects at one time, your site may have established an installation default MEMLIMIT that sets the total number of usable virtual pages above the bar for a single address space. If the default is not sufficient, QMF cannot be started.

The current default for MEMLIMIT for z/OS V1R10.0 and later is 2G. The default for z/OS versions prior to that is 0, meaning that no address space can use virtual storage above the bar. You should verify that the MEMLIMIT setting for your site is at least 2G for QMF. This amount is recommended for scalability of a wide range of objects. QMF only uses as much storage above the bar as it needs, and only as long as it is needed; the storage is released immediately after a job or process is complete.

To set the MEMLIMIT parameter, use one of the following methods:

- Set an installation default on the MEMLIMIT parameter in the SMFPRMxx PARMLIB member.
- Issue the SET SMF or SETSMF command.
- Add either the MEMLIMIT parameter or REGION=0 to the logon procedure TSO JCL or the job JCL.
- Specify MEMLIMIT in a IEFUSI exit routine; if you do so, this MEMLIMIT setting overrides all other MEMLIMIT settings.

The following example shows the MEMLIMIT parameter added to a TSO logon procedure:

```
//TSOPROC EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=175,  
// PARM='%LOGINIT',TIME=1440,MEMLIMIT=2G
```

See the IBM z/OS documentation for your version of z/OS for additional information about MEMLIMIT.

**Note:** As a result of these requirement changes, existing QMF TSO address space allocations might need to be changed.

---

## Chapter 7. Deleting an earlier release of QMF

The procedures for removing an earlier release of QMF depend on the type of database in which QMF is installed. You should remove the release only after you verify you do not need to connect to that release and you have migrated everything you want to save from the release.

### Before you begin

Remove the earlier release of QMF from the database only after you confirm these conditions:

- Connectivity is no longer required with that release. For example, if you are considering removing QMF Version 9.1 from a server, verify that CONNECT commands from Version 9.1 installations in requester databases will no longer be directed to the server.
- You have migrated and tested all programs as well as tables, views, queries, forms, procedures, and other objects that you want to save from the earlier release.

### Related concepts

Migration and fallback considerations

After you install or migrate to QMF Version 12.1, you might need to access and use objects from earlier releases of QMF to thoroughly test the new installation under site-specific conditions.

---

## Deleting QMF from a Db2 for z/OS database

The procedures for removing earlier releases of QMF from a Db2 for z/OS database depend on whether the old and new releases of QMF are in the same subsystems.

### Before you begin

**Important:** Do not begin the tasks in this topic until you have verified that the earlier version of QMF is no longer needed in this database. The earlier version is still needed if it is being accessed locally or if connectivity with remote installations of the same release is required.

### When the old and new release are in the same subsystem

You free the application plan, delete the libraries, and drop all packages from the earlier release of QMF to delete that release. This procedure applies when both releases of QMF on a Db2 for z/OS database are in the same subsystem.

#### Freeing the earlier application plan

You free the application plan of an earlier release of QMF that resides in the same subsystem as the new release by editing the DSQ1DEL1 job and running the DSQ1JFPL jobs.

#### Procedure

1. Edit QMF1210.SDSQSAPE(DSQ1JFPL).
2. Change the job statement to conform to your site's needs.
3. Verify (or change, if necessary) the values of the parameters in the instream procedure of the job.

```
//DSQ1JFPL PROC RGN='2048K',      Job-step region size
//  QMFTPRE='QMF1210',          QMF
//  prefix
//  DB2EXIT='DSN1110.SDSNEXIT',  DB2 for z/OS exit library
//  DB2LOAD='DSN1110.SDSNLOAD'  DB2 for z/OS program library
```

4. Edit QMF1210.SDSQSAPE(DSQ1DEL1).
5. Replace the DSN SYSTEM value that appears in parentheses with the name of the Db2 for z/OS subsystem. Replace the FREE PLAN value with the application plan name of the earlier release.

This table shows default application plan names for releases from which migration to QMF Version 12.1 is supported.

<i>Table 24. Default values for QMF application plan names</i>	
<b>Prior release</b>	<b>Default application plan name</b>
QMF Version 11.2	QMF1120
QMF Version 11.1	QMF11
QMF Version 10.1	QMF10
QMF Version 9.1	QMF910
QMF Version 8.1	QMF810
QMF Version 7.2	QMF720
QMF Version 7.1	QMF710
QMF Version 6.1	QMF610
QMF Version 3.3	QMF330

For example, if you migrated to QMF Version 12.1 from QMF Version 9.1 in a Db2 for z/OS subsystem called DB2P, make the following changes in job DSQ1DEL1 to free the Version 9 plan:

```
DSN SYSTEM(DB2P)
FREE PLAN(QMF910)
```

6. Start the Db2 for z/OS subsystem in which the old and the new releases reside if you have not done so already.
7. Submit job QMF1210.SDSQSAPE(DSQ1JFPL).  
If the job fails, correct any errors and rerun it.

### Using SMP/E to delete the libraries of the earlier release

Use SMP/E to delete the libraries of the release of QMF from which you migrated to Version 12.1. Delete the libraries after you free the earlier application plan.

### About this task

**Important:** Pay special attention to the prefixes to avoid deleting the current libraries. The default prefix for all Version 12.1 libraries is QMF1210.

The following libraries are used in QMF Version 8.1, Version 9.1, Version 10.1, and Version 11.1. For Version 8.1 the prefix is QMF810, for Version 9.1 the prefix is QMF910, for Version 10.1 the prefix is QMF1010, and for Version 11.1 the prefix is QMF1110.

- *prefix*.SDSQCHRT
- *prefix*.ADSQCHRT
- *prefix*.SDSQCLTE
- *prefix*.SDSQDBRM
- *prefix*.ADSQDBMD
- *prefix*.DSQDEBUG
- *prefix*.SDSQEXCE
- *prefix*.SDSQEXIT
- *prefix*.SDSQLOAD
- *prefix*.ADSQMACE
- *prefix*.SDSQMAPE



- *prefix*.ADSQMAPE
- *prefix*.SDSQMLBE
- *prefix*.ADSQOBJ
- *prefix*.SDSQPLBE
- *prefix*.ADSQPMSE
- *prefix*.DSQPNLE
- *prefix*.SDSQPVRE
- *prefix*.ADSQPVRE
- *prefix*.SDSQSAPE
- *prefix*.SDSQSLBE
- *prefix*.SDSQUSRE
- *prefix*.DSQUCFRM

The following table lists the libraries for QMF Version 7.2 and earlier releases.

*Table 25. Names of libraries for QMF Version 7.2 and earlier releases*

Version 3.3 libraries	Version 6.1 libraries	Version 7.1 libraries	Version 7.2 libraries
QMF330.DSQCHART	QMF610.SDSQCHRT	QMF710.SDSQCHRT	QMF720.SDSQCHRT
QMF330.DSQCLSTE	QMF610.SDSQCLTE	QMF710.SDSQCLTE	QMF720.SDSQCLTE
QMF330.DSQDBRM	QMF610.SDSQDBRM	QMF710.SDSQDBRM	QMF720.SDSQDBRM
QMF330.DSQDBRMD	QMF610.ADSQDBMD	QMF710.ADSQDBMD	QMF720.ADSQDBMD
	QMF610.DSQDEBUG	QMF710.DSQDEBUG	QMF720.DSQDEBUG
QMF330.DSQEXECE	QMF610.SDSQEXCE	QMF710.SDSQEXCE	QMF720.SDSQEXCE
			QMF720.SDSQEXIT
QMF330.DSQLOAD	QMF610.SDSQLOAD	QMF710.SDSQLOAD	QMF720.SDSQLOAD
QMF330.DSQMACE	QMF610.ADSQMACE	QMF710.ADSQMACE	QMF720.ADSQMACE
QMF330.DSQMAPE	QMF610.DSQMAPE	QMF710.DSQMAPE	QMF720.DSQMAPE
QMF330.DSQMLIBE	QMF610.SDSQMLBE	QMF710.SDSQMLBE	QMF720.SDSQMLBE
QMF330.DSQOBJ	QMF610.ADSQOBJ	QMF710.ADSQOBJ	QMF720.ADSQOBJ
QMF330.DSQPLIBE	QMF610.SDSQPLBE	QMF710.SDSQPLBE	QMF720.SDSQPLBE
QMF330.DSQPMSE	QMF610.ADSQPMSE	QMF710.ADSQPMSE	QMF720.ADSQPMSE
QMF330.DSQPNLE	QMF610.DSQPNLE	QMF710.DSQPNLE	QMF720.DSQPNLE
QMF330.DSQPVARE	QMF610.DSQPVARE	QMF710.DSQPVARE	QMF720.DSQPVARE
QMF330.DSQSAMPE	QMF610.SDSQSAPE	QMF710.SDSQSAPE	QMF720.SDSQSAPE
QMF330.DSQSLIBE	QMF610.SDSQSLBE	QMF710.SDSQSLBE	QMF720.SDSQSLBE
QMF330.DSQUSERE	QMF610.SDSQUSRE	QMF710.SDSQUSRE	QMF720.SDSQUSRE
QMF330.DSQTLIBE			
QMF330.DSQUCFRM	QMF610.DSQUCFRM	QMF710.DSQUCFRM	QMF720.DSQUCFRM

### Dropping packages from the earlier release

After you delete the libraries of the earlier release, drop all packages from the earlier release of QMF.

### Procedure

1. List all QMF packages for a particular release by running this query.

```
SELECT SUBSTR(NAME,1,10)
FROM SYSIBM.SYSPACKAGE
WHERE NAME LIKE 'DSQX%'
```

In this statement, X is the version-specific character that is associated with the release. This table lists the first four characters of the package names for each release from which migration to QMF Version 12.1 is supported.

<i>Table 26. Package identifiers for releases before QMF Version 12.1</i>	
<b>Release of QMF no longer needed</b>	<b>Package names begin with</b>
Version 3.3	DSQA
Version 6.1	DSQB
Version 7.1	DSQC
Version 7.2	DSQD
Version 8.1	DSQE
Version 9.1	DSQF
Version 10.1	DSQG
Version 11.1	DSQH
Version 11.2	DSQI

2. Issue a DROP statement for each package.

### Example

To list packages that are associated with QMF Version 10, issue the following statement:

```
SELECT SUBSTR(NAME,1,10)
FROM SYSIBM.SYSPACKAGE
WHERE NAME LIKE 'DSQG%'
```

## When the old and new release are in different subsystems

When the old and new releases of QMF are in different Db2 for z/OS subsystems and you delete the earlier release, you remove all traces of QMF and objects created by QMF.

### Before you begin

**Important:** Complete this task only if the old and new releases of QMF are in different subsystems and only if connectivity to the earlier release is no longer required.

### Procedure

1. Migrate to the QMF Version 12.1 subsystem any QMF queries, procedures, or forms from the earlier release that you still want to use in Version 12.1.
2. Delete the libraries of the earlier release of QMF as explained in [“Using SMP/E to delete the libraries of the earlier release”](#) on page 90.
3. Drop the QMF packages of the earlier release as explained in [“Dropping packages from the earlier release”](#) on page 91.
4. Delete the QMF constructs and QMF objects from the earlier release by customizing and running job DSQ1DELA. This job deletes the QMF control tables, including the QMF object control tables and any queries, procedures, or forms that they contain.
  - a) Edit QMF1210.SDSQSAPE(DSQ1DELA).
  - b) Change the job statement to conform to your site's conventions.
  - c) Verify (or change, if necessary) the values of the parameters in the instream procedure of the job:

```
// DSQ1DELA PROC RGN='2048K',      Job-step region size
// QMFTPRE='QMF1210',           QMF prefix
```

```
// DB2EXIT='DSN1110.SDSNEXIT',    DB2 for z/OS exit library
// DB2LOAD='DSN1110.SDSNLOAD'    DB2 for z/OS program library
```

- d) Edit member QMF1210.SDSQSAPE(DSQ1DEL1). This job frees the QMF application plan of the earlier release.

In job DSQ1DEL1, replace the DSN SYSTEM value that appears in parentheses with the name of the Db2 for z/OS subsystem. Replace the FREE PLAN value with the application plan name of the earlier release of QMF. Default QMF application plan names are shown in [Table 24 on page 90](#).

For example, if you migrated to QMF Version 12.1 from QMF Version 9.1 in a Db2 for z/OS subsystem called DB2P, make the following changes in job DSQ1DEL1 to free the Version 9 plan:

```
DSN SYSTEM(DB2P)
FREE PLAN(QMF910)
```

- e) Edit member QMF1210.SDSQSAPE(DSQ1DEL2).

This member contains SQL statements to drop views, table spaces, databases, and storage groups.

If the earlier release did not use a table space for the users' SAVE DATA commands and the IVP, delete the following statement:

```
DROP STOGROUP DSQSGDEF
```

- f) Edit member QMF1210.SDSQSAPE(DSQ1DEL3)

This member contains statements to delete user-managed data sets for the QMF control tables. You do not need to complete this step if your data sets are managed by Db2 for z/OS.

If your data sets are not managed by Db2 for z/OS, change QMFDSN in each statement in the job to the high-level qualifier you used for the release of QMF that you are deleting.

5. Start the subsystem in which the earlier release of QMF resides, and submit job QMF1210.SDSQSAPE(DSQ1DELA).

If the job fails, correct any errors and rerun the job.

### Related concepts

#### [Migration and fallback considerations](#)

After you install or migrate to QMF Version 12.1, you might need to access and use objects from earlier releases of QMF to thoroughly test the new installation under site-specific conditions.

## Deleting QMF from a DB2 for iSeries or Db2 for Linux, UNIX, and Windows database

The procedures differ depending on whether the old and new QMF release are in the same or different databases.

### About this task

The procedures in this topic remove earlier releases of QMF from a Db2 for Linux, UNIX, and Windows or DB2 for iSeries database.

**Important:** Do not begin the tasks in this topic until you have verified that the earlier version of QMF is no longer needed in the server. The earlier version is still needed if connectivity is required to QMF at the same earlier release level.

## When the old and new releases are in the same remote server

When the old and new QMF releases are in the same remote server, remove the earlier QMF release by dropping all QMF packages from the earlier release. QMF control tables and sample tables are still needed for QMF Version 12.1 in this case.

### About this task

Table 26 on page 92 lists the version-specific characters of the package identifiers that are associated with the QMF release from which migration to QMF Version 12.1 is supported.

### Procedure

- To list all QMF packages for a particular release, run the statement that is appropriate for your server. In place of X, substitute the version-specific character of the package identifier that is associated with the QMF release you want to remove.

Follow this statement with a DROP statement for each package you want to remove.

- For Db2 for Linux, UNIX, and Windows servers:

```
SELECT SUBSTR(PKGNAME,1,10)
FROM SYSCAT.PACKAGES
WHERE PKGNAME LIKE 'DSQX%'
```

- For DB2 for iSeries servers:

```
SELECT SUBSTR(PACKAGE_NAME,1,10)
FROM QSYS2.SYSPACKAGE
WHERE PACKAGE_NAME LIKE 'DSQX%'
```

### Example

For example, to list QMF Version 9 packages on a DB2 for iSeries server, issue the following statement:

```
SELECT SUBSTR(PACKAGE_NAME,1,10)
FROM QSYS2.SYSPACKAGE
WHERE PACKAGE_NAME LIKE 'DSQF%'
```

## When the old and new release are in different servers

To delete an earlier version of QMF from an iSeries or Linux, UNIX, and Windows database, perform these tasks.

### Before you begin

Migrate to QMF Version 12.1 any QMF queries, procedures, or forms from the earlier release that you still want to use. See [Chapter 6, “Migration and fallback considerations,” on page 85](#) for information about how to migrate objects.

### About this task

**Important:** The procedure in this topic removes all traces of QMF from the remote server. Complete this task only if the old and new releases are in different servers and only if connectivity to the earlier release of QMF is no longer required.

### Deleting QMF control tables and packages

This procedure deletes the QMF control tables, including the QMF object control tables and any queries, procedures, or forms that they contain.

### About this task

Complete the following procedure for Db2 for Linux, UNIX, and Windows or DB2 for iSeries servers that contain a prior version of QMF.

## Procedure

1. Edit QMF1210.SDSQSAPE(DSQ1EDX1).
2. Change *ssid* to the 4-character name of the Db2 for z/OS requester and change *location* to the name of the server database.
3. Review the comments in the job to determine if you need to customize any other values.
4. Submit job QMF1210.SDSQSAPE(DSQ1EDX1).
5. Check the DSQCDROP job step for a return code of 0 or 4. Review SYSTERM for completion messages. If the return code is not 0 or 4, examine SYSTSPRT and SYSPRINT for error messages. Correct any errors and rerun the job.

## Deleting QMF sample tables

This procedure drops all the QMF sample tables from the server.

## Procedure

1. Edit QMF1210.SDSQSAPE(DSQ1EDX2).
2. Change *ssid* to the 4-character name of the Db2 for z/OS requester and change *location* to the name of the server database.
3. Review the comments in the job to determine if you need to customize any other values.
4. Submit job QMF1210.SDSQSAPE(DSQ1EDX2).
5. Check the DSQCDROP job step for a return code of 0 or 4. Review SYSTERM for completion messages. If the return code is not 0 or 4, examine SYSTSPRT and SYSPRINT for error messages. Correct any errors and rerun the job.

## Deleting QMF from a DB2 for VSE and VM database

---

The procedures differ depending on whether the old and new QMF release are in the same or different databases.

### About this task

The procedures in this topic remove earlier releases of QMF from a DB2 for VSE and VM database.

**Important:** Do not begin the tasks in this topic until you have verified that the earlier version of QMF is no longer needed in the server. The earlier version is still needed if connectivity with QMF installations of the same release is required.

## When the old and new releases are in the same DB2 for VSE and VM server

When the old and new QMF releases are in the same DB2 for VSE and VM server, remove the earlier QMF release by dropping all QMF packages from the earlier release. QMF control tables and sample tables are still needed for QMF Version 12.1 in this case.

### About this task

Table 26 on page 92 lists the version-specific characters of the package identifiers that are associated with the QMF release from which migration to QMF Version 12.1 is supported.

## Procedure

- To list all QMF packages for a particular release, run the following statement.  
In place of X, substitute the version-specific character of the package identifier that is associated with the QMF release you want to remove.

Follow this statement with a DROP statement for each package you want to remove.

```
SELECT SUBSTR(PROGNAME,1,10)
FROM SYSTEM.SYSPROGAUTH
WHERE PROGNAME LIKE 'DSQX%'
```

### Example

For example, to list QMF Version 9 packages from DB2 for VSE and VM server, issue the following statement:

```
SELECT SUBSTR(PROGNAME,1,10)
FROM SYSTEM.SYSPROGAUTH
WHERE PROGNAME LIKE 'DSQF%'
```

## When the old and new release are in different servers

To remove QMF from a DB2 for VM or VSE server, complete these steps.

### About this task

**Important:** The procedure in this topic removes all traces of QMF from a remote VM or VSE server. Complete this task only if the old and new QMF releases are in different servers and only if connectivity to the prior QMF release is no longer required.

### Procedure

1. List the old QMF packages by running the following query. Then drop the resulting package names. Replace X in DSQX with the version-specific character from [Table 26 on page 92](#) that corresponds to the QMF release that you want to remove.

```
SELECT SUBSTR(PROGNAME,1,10)
FROM SYSTEM.SYSPROGAUTH
WHERE PROGNAME LIKE 'DSQX%'
```

For example, to list QMF Version 9 packages from a DB2 for VM or VSE server, issue the following statement:

```
SELECT SUBSTR(PROGNAME,1,10)
FROM SYSTEM.SYSPROGAUTH
WHERE PROGNAME LIKE 'DSQF%'
```

2. List views that are used in QMF operations by running the following query. Then drop the resulting view names.

```
SELECT * FROM SYSTEM.SYSVIEWS
WHERE VCREATOR = 'Q'
```

3. List QMF control tables by running the following query. Then drop the resulting table names.

```
SELECT * FROM SYSTEM.SYSCATALOG
WHERE CREATOR = 'Q'
```

4. List dbspaces that are used by QMF by running the following query. Then drop the resulting dbspace names.

```
SELECT * FROM SYSTEM.SYSDBSPACES
WHERE DBSPACENAME LIKE 'DSQ%'
```

---

## Chapter 8. Installing extra value features

After you have completed the base installation of QMF, you can install extra value features.

### Configuring QMF Analytics for TSO

---

Configuring QMF Analytics for TSO involves customizing and allocating files.

#### Before you begin

Before using QMF Analytics for TSO, you must:

- Install, configure, and run the installation verification procedure (IVP) for the IBM Graphical Data Display Manager - PGF (GDDM-PGF).
- Ensure that the emulator that QMF Analytics for TSO will access can display data graphically (charts and statistical graphs).
- Complete the QMF Analytics for TSO V12 installation and IVP.

#### Load library considerations for QMF Analytics for TSO

The modules that are needed for QMF Analytics for TSO are prefixed with "DYQ" and exist in the QMF for TSO load library, SDSQLOAD.

The load library method of allocation using DSQLLIB cannot be used for SDSQLOAD if you use QMF Analytics for TSO.

#### Allocating files used by QMF Analytics for TSO

In order to use QMF Analytics for TSO you must allocate the standard QMF for TSO files and the QMF Analytics for TSO configuration file.

#### About this task

The QMF Analytics for TSO configuration file is `hlq.SDYQCFG(DYQCFG)`, with a DDNAME of `DYQCFG`. Allocating it results in the default configuration for QMF Analytics for TSO.

#### Procedure

You can allocate DDNAME `DYQCFG` in one of the following ways:

- From a QMF procedure or QMF command line by using a command similar to the following command:  
**`TSO ALLOC FI(DYQCFG) DSN('QMF1110.SDYQCFG(DYQCFG)')`**
- Along with your standard QMF allocations in REXX :  
**`"ALLOC FI(DYQCFG) DA('QMF1110.SDYQCFG(DYQCFG)') SHR REUSE"`**

#### Results

You have allocated the QMF Analytics for TSO configuration file.

#### What to do next

If the default configuration does not suit your environment, you can customize it.

## File system customization for QMF Analytics for TSO

QMF Analytics for TSO uses zSeries file system (zFS) to store a user's work files. You must provide the space for work files for each user in the zFS directory.

### QMF Analytics for TSO and work files

QMF Analytics for TSO uses work files to manipulate your data while producing charts and statistical analyses. It is important to keep the work files for each user separate to avoid users displaying another each other's data.

The default location for ZFS work file usage is determined by the ':asfiles.ZFS' configuration entry in the hlq.SDYQCFG(DYQCFG) configuration file.

QMF Analytics for TSO keeps user's work files separate by storing its work files in a directory named /QAT/ in the user's home directory on the file system (zFS or HFS file).

You can use this default configuration as long as each user on the system has been set up with a unique home directory. However, if your environment is configured so that users share a common home directory, or if you simply do not want to use the location as specified in the default configuration, you set the directory location into which QMF Analytics for TSO will store its work files. To do this, you edit the :asfiles entry in the QMF Analytics for TSO Configuration file (DYQCFG).

**Important:** The :asfiles. entry is the only line in the QMF Analytics for TSO configuration file that should ever be altered. Altering any other lines or altering the :asfiles. entry other than in accordance with these instructions will result in QMF Analytics for TSO not functioning correctly.

When shipped, the default setting for :asfiles. entry is :asfiles.ZFS. As mentioned, using this setting results in QMF Analytics for TSO storing its work files in [home\_directory]/QAT/.

If you want QMF Analytics for TSO work files to be placed in a different location, edit the :asfiles.ZFS setting as follows:

- Keep the setting as :asfiles.ZFS to have work files stored in [user's\_home\_directory]/QAT/.
- Specify :asfiles.ZFS root (no leading /) to have work files stored in [user's\_home\_directory]/root/QAT/.
- Specify :asfiles.ZFS /root (leading /) to have work files stored in [file\_system\_root]/root/QAT /.

Note that root and /root are limited to 227 characters in length and may include many levels of subdirectory. A trailing slash / is optional and QMF Analytics for TSO will assume one if there is none.

To introduce uniqueness into the path name when using /root or when each user's home directory is not unique, you can use &SYSUID. within the path name. When QMF Analytics for TSO interprets this, it replaces &SYSUID. with the current user's user ID in lower case.

The following example shows how one might customize :asfiles.ZFS.

**Note:** QMF Analytics for TSO also supports hierarchical file system (HFS). If you are using HFS, the same exact setup steps for :asfiles apply.

```
*-----*
* QAT Configuration Entry for use with zFS/HFS      *
*-----*
:nick.QAT      :type.ENV
                :asfiles.ZFS products/workfiles/&SYSUID./QMF
                :library.ZFS
                :log.OFF
                :sortname.ICEMAN
```

With this setting, work files for users 'Alan' and 'FINANCE' are stored as follows:

- For Alan:  
[home\_directory]/products/workfiles/alan/QMF/QAT/
- For FINANCE:



[home\_directory]/products/workfiles/finance/QMF/QAT/

If the path had been specified with a leading / (i.e.: asfiles.ZFS /products/), the /products/ directory would be placed in the file system root directory rather than the user's home directory.

## Installing QMF Analytics for TSO sample tables

You can optionally install QMF Analytics for TSO sample tables, which are referred to in the documentation to demonstrate QMF Analytics for TSO functions. These sample tables are Q.CASHFLOW, Q.CLIMATE\_10YR, Q.CLIMATE\_USA, and Q.WORLDINFO.

### Before you begin

If you installed QMF Analytics for TSO in a Db2 for z/OS or Db2 for Linux, UNIX, and Windows database, install the base QMF sample tables before you install the QMF Analytics for TSO sample tables. In Db2 for z/OS, the job that installs the base sample tables is DSQ1EIVS. In Db2 for Linux, UNIX, and Windows, the job that installs the base sample tables is DSQ1EDJ4. You do not need to install the base sample tables before installing the QMF Analytics for TSO sample tables in a DB2 for iSeries database.

### Procedure

Run job QMF1210.SDSQSAPE(DSQ1EIVQ).

In Db2 for Linux, UNIX, and Windows, job DSQ1EIVQ installs the sample tables in the DSQ1STBT table space, which was created by job DSQ1EDJ4. In Db2 for z/OS, job DSQ1EIVQ creates universal table spaces for each sample table.

## Verifying that QMF Analytics for TSO installed correctly

You can verify that QMF Analytics for TSO is installed correctly and test the product functionality by creating a chart and statistical analysis of query results.

### Before you begin

Complete the following tasks before you verify the installation:

- Install, configure, and run the installation verification procedure (IVP) for the IBM Graphical Data Display Manager - PGF (GDDM-PGF).
- Ensure that the emulator that QMF Analytics for TSO will access can display data graphically (charts and graphs).
- Ensure that you installed QMF for TSO V12.1 and completed the QMF for TSO IVP.

### About this task

The verification procedure involves running queries against two sample data tables that are provided with QMF for TSO:

- Q.STAFF

This table provides personnel-related data such as salary, department, commissions, and hire date for employees.

- Q.CLIMATE\_10YR

This table provides climate and weather-related data such as rainfall amount, days of sunshine, and temperature that is collected over many years.

### Procedure

To verify that QMF Analytics for TSO installed correctly:

1. Start QMF.
2. Enter **DISPLAY Q.STAFF** on the command line.
3. Start QMF Analytics for TSO by entering **SHOW ANALYTICS** on the QMF command line and pressing Enter.

4. On the Charts panel of the **QMF Analytics Home** page, tab to Tower and press Enter.
5. With the cursor in the first data entry field, press the List function key (F4). Move to **COMM** on the **Column Selection** panel and press Enter.  
QMF Analytics for TSO returns you to the Charts panel and inserts COMM in the data entry field.
6. Move to the second data entry field on Tower chart's **Parameter Selection** panel and press the List function key (F4). Move to **SALARY** on the **Column Selection** panel and press Enter.  
QMF Analytics for TSO returns you to the Charts panel and inserts SALARY In the data entry field.
7. With the cursor in the data entry field for the X-axis, press the List function key (F4). Move to **DEPT** on the **Column Selection** panel and press Enter.  
QMF Analytics for TSO returns you to the Charts panel and inserts DEPT in the data entry field.
8. Press the Run function key to create the chart.
9. To return to the **Parameter Selection** panel, press F12.
10. To return to the **QMF Analytics Home** panel, press the End function key (F3).
11. Enter **DISPLAY Q.CLIMATE\_10YR** on the QMF command line.
12. Enter **SHOW ANALYTICS** on the command line.
13. Move to Basic in the **Statistics** section of the **QMF Analytics Home** page and press Enter.
14. With the cursor in the first data entry field, press the List function key (F4).
15. Move to **RAINFALL** on the **Column Selection** panel and press Enter.  
QMF Analytics for TSO returns you to the Statistics panel and inserts RAINFALL in the data entry field.
16. Press the Run function key (F2) to see a summary of the basic statistics for rainfall.
17. Press the Statistics function key (F10) to generate the basic statistics of rainfall data.

## Configuring QMF Data Service (QDS)

QMF for z/OS and QMF Enterprise Edition users can use the QMF Data Service feature to access non-Db2 data such as VSAM, IMS, sequential files, SMF data, SYSLOG data and more.

### Before you begin

The QMF Data Services component, available to QMF for z/OS and QMF Enterprise Edition users, must be installed and available to the QMF for TSO and CICS environment. Data sources must be defined to QMF Data Service. The QMF for TSO and CICS global variable DSQEC\_DS\_SUPPORT must be set to a value of 1. See the *IBM Db2 QMF Data Service Getting Started Guide* for more complete instructions on installing the QDS component and configurations including binding packages and allocating proper QDS libraries. These tasks must be performed for the QDS and QMF for TSO and CICS configuration to be complete.

### About this task

Through QMF for TSO/CICS SQL queries, Prompted queries, or Query-by-Example queries, you can access QMF Data Service defined data sources using three part names; QMF Data Service data sources are accessed through three part table names in the query.

For example, you might access a VSAM data set defined to a QMF Data Service server named CQDR by issuing the following query:

```
SELECT * FROM CQDR.CQDSQL.VSAM_IMITMTRN
```

QMF Data Service might join one or more sources that exist at the server. SQL accepted by QMF Data Service is a subset of SQL accepted by Db2 for z/OS. Refer to the QMF Data Service SQL guide for accepted SQL syntax.

To use QMF Data Service, you must set the QMF for TSO and CICS global variable DSQEC\_DS\_SUPPORT to 1. The default value is 0, which indicates that you do not want to allow access to QMF Data Service.

The DISPLAY TABLE, DRAW, EXPORT TABLE, and PRINT TABLE commands may also be used to access QMF Data Service data sources. As with queries, the command must use a three part table name.

For example, you might access a VSAM data set defined to the QMF Data Service server CQDR by issuing the following command:

```
DISPLAY TABLE CQDR.CQDSQL.VSAM_IMITMTRN
```

When you set the DSQEC\_DS\_SUPPORT global variable to a value of '1', the QMF Data Service component analyzes the command. If an object that is referenced in the command is defined to the QMF Data Service component, then the entire command is executed by QMF Data Service. If none of the objects referenced in the command access an object defined to the QMF Data Service, then the command is executed by the current DB2 connection.

## Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### Before you begin

- If you have not done so already, install, prepare, and test QMF Version 12.1 (English) in the database into which you intend to install the NLF. This installation is the base product.
  - For installation paths for requesters, see [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)”](#) on page 14.
  - For installation paths for servers that will be accessed by the QMF CONNECT command, see [“Roadmap 2: Installing QMF in server databases accessed by the QMF CONNECT command”](#) on page 19.
  - If the server will be accessed by three-part names only, you do not need to perform any further NLF-related steps if you already completed [“Installation path G: Preparing a remote server to be accessed by QMF commands that include three-part names”](#) on page 28.
- Ensure that you have the proper authority for the database into which you will be installing the NLF. For more information, see [“Required authorities for installing QMF”](#) on page 13.

### About this task

Names of some modules, libraries, and jobs contain an *n* character. This character is a variable that represents the 1-character language identifier for your NLF. Substitute the language ID associated with your NLF wherever you see this character. You can find these identifiers in [Table 27 on page 102](#).

### Procedure

To install a QMF NLF, complete the following steps:

1. Read the NLF program directory for supplementary data.

There are several program directories. Make sure that you refer to the National Language Version program directory that is appropriate for your edition of QMF. The program directories are updated between NLF releases and they contain useful information, including descriptions of PTFs and APARs.

2. Use SMP/E, as explained in the program directory, to load the NLF libraries for QMF Version 12.1 from the distribution media into the database into which you installed the base product.

Ensure that you complete all the SMP/E tasks that are described in the NLF program directory for each NLF that you intend to install.

The JCL and control statements for the NLF are shipped on the IBM software distribution (ISD) tape for that feature.

The following table provides information about each language in which QMF is available. The function modification identifier (FMID) identifies each QMF NLF to SMP/E.

The 1-character language identifier (language ID) is used in data set, library, and module names. QMF uses the translated names in column 2 in certain QMF control tables. English is included for completeness only; English is not a National Language Feature.

The NLF installation requires the use of the sample libraries from QMF Version 12.1, QMF1210.SDSQSAPE and QMF1210.SDSQSAPn, and the load module library, QMF1210.SDSQLOAD.

National Language Feature	Name that QMF uses for this NLF	Language ID	QMF Version 12.1 FMID
English	ENGLISH	E	HSQCC10
U/C English	UPPERCASE	U	JSQCC51
Danish	DANSK	Q	JSQCC55
Canadian French	FRANCAIS CANADIEN	C	JSQCC5G
French	FRANCAIS	F	JSQCC56
German	DEUTSCH	D	JSQCC57
Italian	ITALIANO	I	JSQCC58
Japanese	NIHONGO	K	JSQCC59
Korean	HANGEUL	H	JSQCC5A
Brazilian Portuguese	PORTUGUES	P	JSQCC5B
Spanish	ESPAÑOL	S	JSQCC5C
Swedish	SVENSKA	V	JSQCC5D
Swiss French	FRANCAIS (SUISSE)	Y	JSQCC5E
Swiss German	DEUTSCH (SCHWEIZ)	Z	JSQCC5F

SMP/E associates all modifications of a program with a system release level (SREL) of that program. The system release level for QMF is P115.

3. Start the database into which you will be installing the NLF.

If you are installing the NLF into a database that will function as a server, DRDA communications between the server and the requester will be used to complete the NLF installation of QMF Version 12.1 in the server. Therefore, you must complete these steps:

- a) Define and test DRDA communications between the server into which you are installing the NLF and the requester into which you installed the base QMF Version 12.1 product.

For more information about how to define and test DRDA, see the Db2 documentation.

- b) Start both the requester and server databases. The Db2 for z/OS subsystem in the requester database must be started with distributed data facility (DDF).

The NLF must be installed into each Db2 subsystem where you will start QMF in that national language. Multiple NLFs can be installed in each subsystem if necessary.

4. Customize the installation parameters in the jobs in Step “6” on page 103 for the NLF that you are installing.

Table 11 on page 40 provides descriptions of the installation parameters in QMF installation jobs. Commonly customized parameters include QMFTPRE, DB2EXIT, DB2LOAD, SSID, and LOCATION. For each job, ensure that the job-step region size on the RGN parameter is 2048K (this setting is the default).

Space allocations for the Q.COMMAND\_SYNONYM\_n table, which stores NLF-specific command synonyms, are set at 100 for the primary space allocation and 20 for the secondary space allocation. The index on the table, Q.COMMAND\_SYNONYMX\_n, has the same primary and secondary allocations.

5. Populate the VSAM panel library for the NLF by running job QMF1210.SDSQSAPn(DSQ1nPNL). For more information about this job, see the QMF program directory.
6. Run the installation jobs in the following table in the order that they are shown for each NLF that you install.

For all steps that run in TSO batch mode, check the step completion code in the system messages. Completion messages can be found in the SYSTSPRT or the SYSTERM output. Check for a return code of 0 or 4.

As each job completes, review SYSTERM for completion messages. If errors occur, examine SYSTSPRT and SYSPPRINT for error messages. SYSPPRINT provides more diagnostic information for IBM Software Support. If any job fails, correct any errors and rerun it.

<i>Table 28. Job sequence for installing a QMF National Language Feature into a requester or server database</i>	
<b>Job name</b>	<b>Description</b>
Each of the following jobs creates the QMF NLF control tables in the target database and adds two rows to the Q.PROFILES table to support the NLF (a row for TSO and a row for CICS). These rows are inserted with a QMF user ID value of SYSTEM in the CREATOR column of the table.	
QMF1210.SDSQSAPn(DSQ1nUPO)	Run this job if you are installing the NLF into a Db2 for z/OS database.
QMF1210.SDSQSAPn(DSQ1nDJ2)	Run this job if you are installing the NLF into a Db2 for Linux, UNIX, and Windows database.
QMF1210.SDSQSAPn(DSQ1nAS2)	Run this job if you are installing the NLF into a DB2 for iSeries database.
QMF1210.SDSQSAPn(DSQ1nCCS)	Creates a command synonym table named Q.COMMAND_SYNONYM_n for the NLF environment.
<p>The following jobs delete existing and install new versions of the QMF NLF sample tables.</p> <p>Prior versions of the NLF sample tables are still valid in the new QMF NLF installation. These jobs are shown in case you must delete and reinstall the samples for any reason. The SELECT privilege is granted to PUBLIC for all sample tables at the location in which you are installing the NLF. You can rerun these jobs, if necessary, at any time.</p> <p>Sample tables are not available for the Uppercase English or Swedish NLFs.</p>	
QMF1210.SDSQSAPn(DSQ1nDSJ)	Deletes existing QMF NLF sample tables from Db2 for z/OS databases.
QMF1210.SDSQSAPn(DSQ1nIVS)	<p>Installs QMF NLF sample tables on Db2 for z/OS databases. The CDS and CDP parameters identify the punctuation mark to use for the decimal point. This value must match the DECPOINT option that was specified when Db2 was installed:</p> <ul style="list-style-type: none"> <li>• For a period, leave the current values as they are.</li> <li>• For a comma, change CDS to 6 and CDP to 7.</li> </ul>

Table 28. Job sequence for installing a QMF National Language Feature into a requester or server database (continued)

Job name	Description
QMF1210.SDSQSAPn(DSQ1nDX2)	Deletes existing QMF NLF sample tables from the following databases: <ul style="list-style-type: none"> <li>• DB2 for iSeries servers</li> <li>• Db2 for Linux, UNIX, and Windows servers</li> </ul> To delete sample tables from VM or VSE databases, issue DROP statements for any tables that you want to remove.
QMF1210.SDSQSAPn(DSQ1nAS4)	Installs QMF NLF sample tables on DB2 for iSeries databases.
QMF1210.SDSQSAPn(DSQ1nDJ4)	Installs QMF NLF sample tables on Db2 for Linux, UNIX, and Windows databases.

7. If you want to install the QMF Analytics for TSO sample tables, run the QMF1210.SDSQSAPn(DSQ1nIVQ) job.

The DSQ1nIVQ jobs are supplied with the corresponding QMF NLF.

**Important:** For Db2 for z/OS or Db2 for Linux, UNIX, and Windows servers, run the DSQ1nIVS or DSQ1nDJ4 job to install the base QMF sample tables before running job DSQ1nIVQ.

8. Customize the QMF NLF for the environment in which it is running:

Option	Description
<b>If QMF is running in TSO</b>	<p>Edit the TSO logon procedure for the QMF base (English) product that you prepared in “Preparing the TSO logon procedure” on page 50. Concatenate each NLF statement that is highlighted here to each English statement in the logon procedure.</p> <pre> //ADMGGMAP DD DSN=QMF1210.SDSQMAPn,DISP=SHR //          DD DSN=QMF1210.SDSQMAPE,DISP=SHR //ISPPLIB DD DSN=QMF1210.SDSQPLBn,DISP=SHR //          DD DSN=QMF1210.SDSQPLBE,DISP=SHR //ISPMLIB DD DSN=QMF1210.SDSQMLBn,DISP=SHR //          DD DSN=QMF1210.SDSQMLBE,DISP=SHR //ISPTLIB DD DSN=QMF1210.SDSQTLBn,DISP=SHR //          DD DSN=QMF1210.SDSQTLBE,DISP=SHR //          DD DSN=ISP.SISPTENU,DISP=SHR //SYSPROC DD DSN=SYS2.CLIST,DISP=SHR //          DD DSN=QMF1210.SDSQCLTn,DISP=SHR //          DD DSN=QMF1210.SDSQCLTE,DISP=SHR //SYSEXEC DD DSN=SYS2.EXEC,DISP=SHR //          DD DSN=QMF1210.SDSQEXCn,DISP=SHR //          DD DSN=QMF1210.SDSQEXCE,DISP=SHR //DSQPNLn DD DSN=QMF1210.DSQPNLn,DISP=SHR //          DD DSN=QMF1210.DSQPNLE,DISP=SHR </pre>
<b>If QMF is running in CICS</b>	<p>a. Load the QMF/GDDM maps to the GDDM ADMF data set by running job DSQ1nADM. Change the parameters QMFTPRE, GDDMADM, and RGN as necessary to suit the NLF that you are installing. This job is explained for the base product in <a href="#">“Loading QMF GDDM maps to the GDDM ADMF data set”</a> on page 58.</p> <p>b. Customize and run job DSQ1nCSD to create entries for the QMF NLF in the CICS system definition file. This job is explained for the base product in <a href="#">“Describing QMF to CICS”</a> on page 56.</p> <p>The return code for this job can be 0 or 4.</p>

Option	Description
	<p>c. Add a line that allocates the QMF NLF panel file to the existing JCL that is used to start the CICS region that contains QMF. This JCL is shown in <a href="#">“Updating the CICS startup job stream”</a> on page 59.</p> <p>To add a line for the QMF NLF panel file, first locate the statement for the QMF base product in the CICS startup JCL:</p> <pre>//DSQPNLE DD DSN=QMF1210.DSQPNLE,DISP=SHR</pre> <p>Then, before that statement, add the following statement for the QMF NLF panel file:</p> <pre>//DSQPNLn DD DSN=QMF1210.DSQPNLn,DISP=SHR</pre>

9. Start QMF under TSO by using the procedure in [“Starting QMF under TSO”](#) on page 72.

10. Install the NLF versions of the QMF sample queries and procedures by deleting the existing NLF queries and procedures, if necessary, and then creating the Version 12.1 NLF objects.

- a) To delete existing NLF versions of the sample queries and procedures, issue the following commands from the QMF for TSO command line. If you are using a site-specific QMF library prefix rather than the default prefix of QMF1210, replace QMF1210 in the command and throughout each procedure before you save it.

```
IMPORT PROC FROM QMF1210.SDSQSAPn(DSQ1nSQD) '
SAVE PROC AS Q.DSQ1nSQD
RUN PROC
```

- b) Create the Version 12.1 NLF sample queries and procedures by issuing the following commands:

```
IMPORT PROC FROM 'QMF1210.SDSQSAPn(DSQ1nSQI)
SAVE PROC AS Q.DSQ1nSQI
RUN PROC
```

For example, to delete the existing German sample queries and procedures, issue the following commands:

```
IMPORT PROC FROM QMF1210.SDSQSAPD(DSQ1DSQD) '
SAVE PROC AS Q.DSQ1DSQD
RUN PROC
```

To create the Version 12.1 German sample queries and procedures, issue the following commands:

```
IMPORT PROC FROM 'QMF1210.SDSQSAPD(DSQ1DSQI)
SAVE PROC AS Q.DSQ1DSQI
RUN PROC
```

If a failure occurs during this job, correct the error and rerun the DSQ1nSQD procedure. Then, rerun the DSQ1nSQI procedure. For more information about the sample queries and procedures, see [“Installing sample queries and procedures required for the installation verification procedures”](#) on page 77.

11. Test for the correct version of the QMF NLF panel library by using the procedure in [“Checking for the correct version of the QMF panel library”](#) on page 75. Verify that both the home panel and help panels display in the language you installed.
12. Run the installation verification procedure appropriate for your environment. If you will run QMF under CICS, complete the IVP for TSO first and then complete the IVP for CICS.

Option	Description
<b>For QMF NLFs under TSO</b>	Issue the following commands from the QMF command line to run the IVP. If necessary, customize the procedure for your site's requirements. For example, if you are using a library

Option	Description
	<p>prefix other than the default of QMF1210, replace QMF1210 throughout the installation verification procedure before you save it.</p> <pre data-bbox="370 275 805 344"> DISPLAY Q.DSQ1nIVP SAVE PROC AS Q.DSQ1nIVP (SHARE=YES RUN PROC </pre>
<b>For QMF NLFs under CICS</b>	<p>Start CICS by using the procedure in “Starting QMF under CICS” on page 75. Then, issue the following commands from the QMF command line to run the IVP. If necessary, customize the procedure for your site's requirements. For example, if you are using a library prefix other than the default of QMF1210, replace QMF1210 throughout the installation verification procedure before you save it.</p> <pre data-bbox="370 569 805 638"> DISPLAY Q.DSQ1nIVC SAVE PROC AS Q.DSQ1nIVC (SHARE=YES RUN PROC </pre>

For more information about the functions that are tested by the IVPs, see one of the following topics:

- [“Running the IVPs for TSO” on page 80](#)
- [“Running the IVP for CICS” on page 82](#)

When the IVP runs without errors, the NLF installation is complete. After the interactive IVP completes on TSO, you can optionally run the batch-mode IVP. For more information, see [“Testing batch operation under TSO” on page 81](#).

13. After installation, delete the libraries of the prior QMF NLF release if necessary. The English versions of these libraries are listed in [“Using SMP/E to delete the libraries of the earlier release” on page 90](#). Replace the E at the end of each name with the national language identifier (NLID) for the NLF you are using.

### What to do next

To change the national language that is displayed during a QMF session, end the current QMF session and begin another. You cannot change the language from within the QMF session.

### Related reference

[About the Q.PROFILES table](#)

Each aspect of a user's QMF session maps to a value in a column of the Q.PROFILES control table. Each row of the Q.PROFILES table is an individual user profile.

### Related information

See the information about QMF Version 12.1 program directories.

## Migrating QMF National Language Features

If you are migrating a QMF installation that includes existing National Language Feature (NLFs), ensure that the correct version of the DSQPNLn data set is allocated, where *n* is the national language identifier. When you migrate to Version 12.1, you must also run a couple of jobs to complete the NLF migration process.

### Procedure

Run job QMF1210.SDSQSAPn(DSQ1nRTS).

This job creates new Version 12.1 command synonyms.



## Installing the QMF stored procedure interface (TSO only)

---

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### Before you begin

If you have not done so already, install, prepare, and test QMF Version 12.1 (English) in the Db2 for z/OS requester database into which you intend to install the interface. For installation paths for requesters, see [“Roadmap 1: Installing QMF in stand-alone or requester databases \(Db2 for z/OS only\)”](#) on page 14. QMF can be started only from a Db2 for z/OS database.

### About this task

Any software program that can call a Db2 for z/OS stored procedure, such as QMF for Workstation, can start QMF for TSO and receive up to 21 result sets back, including a result set for trace data. The name of a predefined procedure or query can be passed to the interface to perform the work and return the required output. The procedure that is passed to the interface can also start a batch job, which allows any Db2 client that can issue a CALL statement to access QMF for TSO batch services.

Similar to QMF batch mode, the stored procedure interface allows no user interaction with QMF; the procedure that is passed to QMF cannot issue commands that display QMF panels (such as LIST, for example).

### Procedure

To install the stored procedure interface:

1. Configure the WLM address space that your site will use to start Db2 for z/OS stored procedures.

The QMF stored procedure interface is made available through a REXX stored procedure, Q.DSQQMFSP, which is external to QMF and must run in a WLM-managed address space that is not APF-authorized.

In general, Db2 for z/OS stored procedures (except native SQL procedures) must run in WLM-managed stored procedure address spaces. You must assign each stored procedure to a WLM application environment, which routes the stored procedure work to the appropriate WLM-managed address space.

Make the following changes to the JCL that starts the WLM-managed address space where QMF will run:

- a) Set the NUMTCB parameter to 1. The Q.DSQQMFSP procedure is a REXX stored procedure, and all REXX stored procedures must be assigned a NUMTCB value of 1.

- b) Specify a region size that meets your site's requirements for running QMF in TSO.

Minimum storage requirements are explained in [“Addressing storage requirements”](#) on page 37. Add to these storage figures the amount of storage that you want to reserve in the TSO region for applications other than QMF; you specify this storage using the DSQSRSTG parameter in the DSQSCMDn exec, as explained in Step [“3”](#) on page 108.

- c) Allocate the QMF1210.SDSQEXCE library to the SYSEXEC ddname.

If you intend to run QMF as a stored procedure in both English and a national language environment, also allocate QMF1210.SDSQEXCn, where n is a 1-character national language identifier.

- d) Allocate the QMF1210.SDSQLOAD load library to the STEPLIB ddname.

- e) Allocate the QMF1210.SDSQMAPE library to the ADMGGMAP ddname.

If you intend to run QMF as a stored procedure in both English and a national language environment, also allocate QMF1210.SDSQMAPn, where n is a 1-character national language identifier.

f) Allocate the following data sets:

**DSQDEBUG**

Receives QMF trace output.

**SYSTSPRT**

Receives REXX trace output when PTF or ALL is specified as the trace option.

- g) Customize GDDM parameters in the same manner as for other QMF for TSO installations.
2. Make sure that the user ID of the user who starts the QMF stored procedure (Q.DSQQMFSP) has read access to the QMF libraries that you allocated in the JCL that starts the WLM address space.
  3. Revise the parameter values in the DSQSCMD $n$  exec, which is located in the QMF1210.SDSQEXC $n$  library. Replace  $n$  with the 1-character national language identifier that represents the language in which you want to start QMF.

The stored procedure program logic uses the DSQSCMD $n$  exec to pass program parameters to QMF for initialization. Within the DSQSCMD $n$  exec, values for the following parameters are required:

**DSQSSUBS**

The name of the subsystem in which you are installing the Q.DSQQMFSP stored procedure.

**DSQSPLAN**

The name of the QMF application plan in the Db2 for z/OS subsystem in which you are installing the Q.DSQQMFSP stored procedure.

The default name for the QMF Version 12.1 application plan is QMF12.

You can optionally set the following program parameters:

**DSQSBSTG or DSQSRSTG**

Amount of virtual storage to use for report operations.

**DSQSMRFI**

Enables multirow processing for database fetches and inserts.

**DSQSPILL and DSQSPTYP**

Enable the use of extra storage to be used for data no longer needed in active storage.

**DSQSDBCS**

Allows you to print DBCS data from non-DBCS display devices.

All other program parameters in the DSQSCMD $n$  exec are either ignored or take only default values when you start QMF as a stored procedure.

4. Run the DSQ1BSP installation job to define the QMF stored procedure.
5. Refresh or resume the WLM address space.
6. Grant the EXECUTE privilege on the Q.DSQQMFSP procedure to all users who will be starting QMF by using this interface.  
For example:

```
GRANT EXECUTE ON PROCEDURE Q.DSQQMFSP TO userid
```

7. Ensure that all users who will start QMF as a stored procedure have a valid QMF profile.

QMF authenticates the authorization ID under which the WLM-managed address space was started with the value in the CREATOR column of the Q.PROFILES table. QMF first checks the CREATOR column for the authorization ID under which the address space was started. If no value in the CREATOR column corresponds to this ID, QMF checks for a row in the Q.PROFILES table where CREATOR=SYSTEM. If no SYSTEM row is found, QMF initialization ends in an error.

Profile settings that are associated with the authorization ID under which the address space was started (or those in the SYSTEM row, if no specific profile is associated with this user ID) serve as default settings for the duration of the QMF session. Because the trace level is passed on the CALL statement, the initial value of the TRACE option of the profile is ignored. Because no prompting can occur when QMF is running as a stored procedure, a value of YES on the CONFIRM option is also ignored.

8. Verify that the installation of the stored procedure interface completed successfully:

- Issue the following CALL statement from QMF for Workstation or another client program. This statement starts the English version of QMF for TSO:

```
CALL Q.DSQQMFSP('','PTF','','',?)
```

The statement returns a service-related string about the Q.DSQQMFSP procedure. The string has the following format:

```
DSQQMFSP exec level: QMF VnRn WIMnnnnn mm-dd-yyyy DSQAO_QMF_VER_RLS:  
QMFVnRn.n DSQ_PRODUCT_RELEASE: nn Current date and time: dd mmm yyyy hh:mm:ss
```

- If you installed the QMF sample queries you can issue the following CALL statement to test starting QMF as a stored procedure and passing the Q.SAMPLE\_SELECT1 query as the first parameter.

This statement specifies no trace log; however, the message that is returned at the end of the stored procedure run indicates whether the query ran successfully.

```
CALL Q.DSQQMFSP('Q.SAMPLE_SELECT1','','',?)
```

To receive a trace log back as a result set, issue the following statement:

```
CALL Q.DSQQMFSP('Q.SAMPLE_SELECT1','L2','','',?)
```

## Results

The DSQAO\_STO\_PROC\_INT global variable records whether QMF was started as a Db2 for z/OS stored procedure.

## Related concepts

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

[Allocating the trace data set](#)

Make sure that the DSQDUMP data set or DSQDEBUG data set is allocated, depending on how your run QMF.

## Related tasks

[Customizing GDDM external defaults](#)

If you are running QMF for TSO, you must customize some external defaults to ensure that GDDM correctly displays information from the database.

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Installing the enhanced LIST command function (z/OS only)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.

### About this task

The following tables and views are displayed in the object list when OWNER is set to ALL:

- Tables and views that are owned by the user's primary or secondary authorization ID

- Tables and views for which SELECT and other privileges have been granted to the user's primary or secondary authorization ID
- Tables and views for which privileges have been granted to PUBLIC

RACF group names can be used as secondary authorization IDs.

The enhanced LIST command function can be used only with Db2 for z/OS databases and must be installed in every database (both requesters and servers) where the function is needed. This topic explains how to install and activate the User Defined Function (UDF) for the feature, which is supplied with QMF.

## Procedure

To install and activate the enhanced LIST command function, complete the following steps:

1. Set up the environment for UDFs. This step involves setting up and maintaining the environment for Db2 stored procedures and UDFs in WLM-established address spaces. A system administrator usually performs this step.
2. Add the QMF program DSQABA1E to the WLM-established address space that will execute the UDF supplied by QMF. DSQABA1E resides in the QMF load library QMF1210.SDSQLOAD. Copy the DSQABA1E member from SDSQLOAD into a load library in the STEPLIB concatenation for the WLM-established stored procedure address. This STEPLIB concatenation is defined in the JCL that is used to start the address space.
3. Determine the name of the WLM environment where the UDF supplied by QMF will run.
4. Define the QMF UDF to Db2 by running the DSQ1BUDF job, which is located in the QMF1210.SDSQSAPE library. This job issues an SQL CREATE FUNCTION statement and grants execution privileges. You might need to customize this job before running it. See the comments in the job prolog for instructions.
5. Test the registration.

Verify that all the previous steps were successful before changing the QMF list view in the next step. To test the registration, start QMF or SPUFI and run the following SQL statement:

```
SELECT U.AUTHNAME FROM TABLE( Q.APPL_AUTHNAMES( 'PUBLIC "PUBLIC*" ' ) ) U
```

The database returns a list of valid authorization names for the user who issued the statement. For example:

```
AUTHNAME
-----
W397754
#DQZA
#J49A
DB2FUNC
QMFDEV
PUBLIC
PUBLIC*
```

6. Change the QMF list view to execute the QMF UDF by running job DSQ1BUDV, which is located in the QMF1210.SDSQSAPE library.
7. Optional: If you customized the views that QMF uses to generate table and view list, examine the following sample SELECT statement, which has been modified to use the UDF supplied by QMF. This example will help you in modifying your customized view:

```
SELECT
  VARCHAR(RTRIM(T.CREATOR)),
  T.NAME, 'TABLE', T.TYPE
  VARCHAR(RTRIM(SUBSTR(T.REMARKS, 1, 254))),
  VARCHAR(RTRIM(SUBSTR(T.LABEL, 1, 30))),
  SUBSTR(T.LOCATION, 1, 16),
  VARCHAR(RTRIM(T.TBCREATOR)), T.TBNAME
FROM SYSIBM.SYSTABLES T
  , ( SELECT
      DISTINCT
      VARCHAR(RTRIM(TA.TCREATOR)), TA.TTNAME
```

```

FROM SYSIBM.SYSTABAUTH TA
WHERE TA.GRANTEETYPE=' '
AND TA.GRANTEE IN (

SELECT U.AUTHNAME FROM
TABLE(Q.APPL_AUTHNAMES
('PUBLIC "PUBLIC*"')) AS U

)
) AS UAT ("CREATOR","NAME")
WHERE
(T.CREATOR=UAT.CREATOR OR T.CREATOR=CURRENT SCHEMA)
AND T.NAME=UAT.NAME
AND T.TYPE IN ('T', 'V', 'H');

```

8. Test the new function.

- a) Grant privileges for tables or views to the primary authorization ID of a particular QMF user. Grant privileges for a different set of tables to this user's secondary authorization ID.
- b) Start QMF under the primary authorization ID.
- c) Issue the command LIST TABLES (OWNER=ALL.  
 You can specify ALL as the default for the OWNER option of the LIST command by setting the DSQEC\_LIST\_OWNER global variable to this value.  
 The list should include all tables for which you granted privileges.

**Results**

The following table summarizes the jobs that install the enhanced LIST command function. Run these jobs in the sequence in which they are listed in the table and ensure that each job completes without errors.

<i>Table 29. Job sequence to install enhanced LIST command function</i>	
<b>Job name</b>	<b>Description</b>
DSQ1BUDF	Creates the enhanced QMF LIST command function. Run this job at every Db2 for z/OS requester or server where the function is needed. Rerun this job when you upgrade the database mode or release level.
DSQ1BUDV	Creates QMF views needed to support the enhanced LIST command. Run this job at every Db2 for z/OS requester or server where the function is needed. Rerun this job when you upgrade the database mode or release level.

**Note:** The installation option EXTSEC found in QMF1210.SDSQEXCE(DSQ1DEFS) supports external security models such as RACF, but does not affect the views created in QMF1210.SDSQEXCE(DSQ1BUDV).

**What to do next**

You can revert to the default QMF LIST command function at any time by running job DSQ1BVW, which restores the views that support the default LIST command.

**Related concepts**

[Making your new view the default view](#)

To use a view that you have created instead of the default view, use a SET GLOBAL command to set the appropriate global variable to the new view name.

[Tasks to perform when upgrading Db2 for z/OS but not QMF](#)

When you upgrade either the mode or the release of a Db2 for z/OS database where QMF is installed, but you do not upgrade QMF, you must run two QMF bind jobs.

### Related tasks

#### Customizing users' object lists

Using the default views supplied by QMF for your table lists and column information can increase processing time, because Db2 gathers authorization information from the SYSIBM.SYSTABAUTH table. If you do not need the extra security provided by these authorization checks, consider creating your own views that generate a list of objects stored in the database.

## Configuring the QMF Data Service Interface

QMF for z/OS V12 and QMF Enterprise Edition V12 adds a new component called QMF Data Service. QMF for TSO and CICS users can use QMF Data Service to access Db2 and non-Db2 data such as VSAM, IMS™, sequential files, SMF data, SYSLOG data and more. See the Db2 QMF Data Service Solution Configuration Guide for a complete list of data sources that can be accessed.

### Before you begin

- Install, configure and run the installation verification procedure (IVP) for QMF Data Service. See the Db2 QMF Data Service Customization Guide for more information.
- For QMF for TSO users, install, configure and run the installation verification procedure for QMF for TSO V12.1 in at least one Db2 for z/OS requester.
- For QMF for CICS users, install, configure and run the installation verification procedure for QMF for CICS V12.1.

### Procedure

For Customizing the QMF for TSO and CICS connection for QMF Data Service involves binding an additional package, allocating load libraries and in the case of CICS use, adding the QMF Data Service program definitions to an existing QMF for CICS region.

1. Bind the database package required for QMF Data Service access. The package must be bound in all Db2 for z/OS systems from which you want to connect to a QMF Data Service server through QMF for TSO or QMF for CICS.
2. **For QMF for TSO implementations only:** Update an existing QMF for TSO environment to add the QMF Data Service library, CQD.SCQDLOAD to either STEPLIB or ISPLLIB.
3. **For QMF for CICS implementations only:** The following steps for defining CICS program definitions and updating DFHRPL should be added to an existing QMF for CICS region that has been successfully installed and tested.
  - Customize the job found in QMF1210.SDSQSAPE(DSQ1ECSQ) according to the comments found in the beginning of the job. Submit the job to successful completion.
  - Add the QMF Data Service library, CQD.SCQDCLOD, to the DFHRPL concatenation. For example:

```
//DFHRPL DD ...
//      DD DSN=QMF1210.SDSQEXIT,DISP=SHR
//      DD DSN=QMF1210.SDSQLOAD,DISP=SHR
//      DD DSN=GDDM.SADMMOD,DISP=SHR
//      DD DSN=DSN1210.SDSNEXIT,DISP=SHR
//      DD DSN=DSN1210.SDSNLOAD,DISP=SHR
//      DD DSN=CQD.SCQDCLOD,DISP=SHR
```

4. Verify the QMF Data Service connection. To begin the verification, start QMF.

**Note:** Ensure that the sample virtual table *staffvs* exists in the QMF Data Service server. Table *staffvs* would have been created during installation and verification of the QMF Data Service server. See the *Verifying the server installation* section of the *Db2 QMF Data Service Customization Guide* for more information on *staffvs*.

5. Enter the following command on the command line:

```
SET GLOBAL (DSQEC_DS_SUPPORT=1
```

6. Run the following SQL query where *cdqs* is the name of the QMF Data Service server:

```
SELECT * FROM cdqs.CQDSQL.STAFFVS
```

A report similar to the one below displays.

```
REPORT                                LINE 1    POS 1    159
                                     STAFFVS  STAFFVS  STAFFVS  STAFFVS  STAFFVS
                                     DATA    DATA    DATA    DATA    DATA
                                     NAME     NAME     DEPT     JOB      YRS
-----
      10      7 SANDERS      20      MGR      7
      20      6 PERNAL      20      SALES    8
      30      8 MARENGHI   38      MGR      5
      40      7 O'BRIEN    38      SALES    6
      50      5 HANES      15      MGR     10
      60      7 QUIGLEY   38      SALES    0
      70      7 ROTHMAN   15      SALES    7
      80      5 JAMES     20      CLERK    0
      90      7 KOONITZ   42      SALES    6
     100      5 PLOTZ     42      MGR      7
     110      4 NGAN      15      CLERK    5
     120      8 NAUGHTON  38      CLERK    0
     130      9 YAMAGUCHI 42      CLERK    6
     140      5 FRAYE    51      MGR      6
     150      8 WILLIAMS  51      SALES    6
     160      8 MOLINARE  10      MGR      7
     170      8 KERMISCH  15      CLERK    4
     180      8 ABRAHAMS  38      CLERK    3
     190      7 SNEIDER   20      CLERK    8
     200      8 SCOUTTEN  42      CLERK    0
     210      2 LU       10      MGR     10
     220      5 SMITH     51      SALES    7
     230      9 LUNDQUIST 51      CLERK    3
     240      7 DANIELS  10      MGR      5
     250      7 WHEELER   51      CLERK    6
     260      5 JONES     10      MGR     12
     270      3 LEA      66      MGR      9
     280      6 WILSON    66      SALES    9
     290      5 QUILL     84      MGR     10
     300      5 DAVIS     84      SALES    5
     310      6 GRAHAM    66      SALES   13
     320      8 GONZALES  66      SALES    4
     330      5 BURKE     66      CLERK    1
     340      7 EDWARDS   84      SALES    7
     350      6 GAFNEY    84      CLERK    5

*** END ***

1=Help      2=          3=End      4=Print    5=Chart    6=Query
7=Backward  8=Forward   9=Form     10=Left    11=Right   12=
OK, this is the REPORT from your RUN command.
COMMAND ==>
```

For more information on using QMF Data Service, see [Configuring QMF Data Service \(QDS\)](#).





---

# Part 2. Managing QMF for TSO and CICS



---

## Chapter 9. Starting QMF

QMF can be started only from z/OS. QMF can be set up to run under TSO, ISPF, as a batch job, or under CICS.

### Starting QMF on TSO

---

There are several ways in which you can start QMF under TSO: You set up QMF to be started on TSO by allocating files and resources and adding CLIST and EXEC libraries to TSO.

#### Allocating necessary files and resources

You can provide new QMF users with a TSO logon procedure that is called when the user logs on. This cataloged procedure calls the Terminal Monitor Program (TMP).

The TMP is the principal interface between the user and the display device during a TSO session. If your site uses its own TMP rather than the procedure supplied by IBM, some of the information in this topic might not apply.

You can develop CLISTs or execs that users run to start QMF. Within these CLISTs or execs, you can allocate many of the required data sets through TSO ALLOCATION statements. You can also use TSO FREE statements in a CLIST or exec to deallocate data sets after the QMF session terminates.

To create a TSO exec to start QMF, you must ensure that the program load libraries, modules, and data sets are available to QMF, and that GDDM and Db2 requirements are met.

#### Adding QMF CLIST and EXEC libraries to TSO

Use the DD statement established by your site for the TSO search order for execs.

This search order is affected by settings in the TSO default modules IRXTSPRM and IRXISPRM, the TSO EXECUTIL command, and the TSO ALTLIB command. These are the data sets used by TSO. If you do not know your site's search order for REXX execs, allocate SDSQEXCE to both SYSEXEC and SYSPROC.

---

```
//*****  
//*      DATA SETS USED BY TSO      *  
//SYSPROC DD DSN=SYS2.CLIST,DISP=SHR      * CLIST Library  
//      DD DSN=QMF1210.SDSQCLTE,DISP=SHR  
//SYSEXEC DD DSN=SYS2.EXEC,DISP=SHR  
//      DD DSN=QMF1210.SDSQEXCE,DISP=SHR
```

Figure 16. Data sets used by TSO

---

#### Starting QMF with the TSO CALL command

To use the TSO CALL command to start QMF, specify the name of the QMF load library and pass optional program parameters following the data set name.

This example starts QMF with the TSO CALL command:

```
CALL 'QMF1210.SDSQLOAD(DSQMF)' 'DSQSMODE=I,DSQSSUBS=DB2T'
```

The QMF load library is allocated as a task library for the duration of the CALL command. However, you must give QMF access to the Db2 and GDDM libraries to load program interfaces to those products. In most cases, Db2 and GDDM libraries are not part of the TASKLIB. If Db2 and GDDM libraries are not available, QMF terminates with an error.

**Important:** Do not use the TSO CALL command from a CLIST that is running under ISPF. When QMF is started under ISPF, you must use the ISPF SELECT PGM method to start QMF.

## Related concepts

### [Starting QMF on ISPF](#)

You can let users start QMF using ISPF services.

## Starting QMF directly with the DSQQMFE module

You can start QMF under TSO by entering DSQQMFE from the command line in READY mode or in a CLIST or exec.

For example, the following command shows how to start and pass parameters to QMF operating independently of ISPF:

```
DSQQMFE DSQSBSTG=50000,DSQSMODE=B
```

The parameters in this example:

- Pass a value of 50,000 for the DSQSBSTG parameter (maximum storage for reports)
- Specify a value of B (batch) for DSQSMODE (mode of operation)

To start from a CLIST or exec and specify an initial procedure, issue a command like the following:

```
DSQQMFE DSQSRUN=Q.IPROC (&&TABLE=Q.STAFF)
```

This statement uses the DSQSRUN parameter to:

- Specify an initial procedure (Q.IPROC) to run when QMF starts
- Pass a value (Q.STAFF) to the procedure for the variable &TABLE

The DSQSRUN parameter as specified in this example results in the following QMF command:

```
RUN Q.IPROC (&TABLE=Q.STAFF
```

When QMF is started in TSO independently of ISPF, the following return codes are valid:

**0**

Execution successful.

**4**

Warning condition occurred.

**8**

Error condition occurred.

**16**

Severe error occurred. QMF did not start. If no messages are displayed, check the trace output for messages, which can help you diagnose the problem.

## Related concepts

### [Viewing QMF trace data](#)

DSQDEBUG holds the information that is recorded by the trace facility. It must be allocated before you start QMF if you want to use tracing.

## Related reference

### [Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## Starting QMF in a batch environment

To start QMF without using ISPF services, place a statement in the SYSTSIN data set of the JCL for the batch job.

For example, in this command, DSQSMODE=B establishes the appropriate operating mode, and the DSQSRUN parameter identifies the owner and name of the procedure to be run.

```
DSQQMFE DSQSMODE=B,DSQSRUN=auth_id.proc_name
```

The statement can include other parameter values in addition to the required DSQSMODE and DSQSRUN parameters.

### Related concepts

#### Running QMF in batch mode

If a user runs a procedure with the RUN command, the user cannot execute QMF commands except to cancel the procedure or the session. Thus, running a procedure using the RUN command can tie up considerable session time.

#### Setting program parameters and preferences at startup time

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

## Starting QMF as a Db2 for z/OS stored procedure

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

Instructions about installing the components of the interface and verifying that it is working correctly are described in the steps to install the stored procedure interface. Similar to QMF batch mode, no panels can be displayed when QMF is running as a stored procedure, so any QMF procedure that runs after QMF starts cannot issue commands such as LIST or DISPLAY.

### Related tasks

#### Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

#### Starting a QMF batch job from a remote Db2 client

Any Db2 client that is connected to the Db2 database where the QMF stored procedure DSQQMFSP is installed can start a QMF for TSO batch job. The batch job is started from the z/OS system that runs the QMF stored procedure.

### Format of the CALL statement

This topic explains the format of the required CALL statement.

The CALL statement that starts QMF for TSO as a Db2 for z/OS stored procedure has the following format:

```
➤➤ CALL — Q.DSQQMFSP —>
```

```
➤ ( ——————>
  |   |   |   |   |   |
  | 'object-name' — 'trace-level' — 'L2-destination' — 'language' — status-message — (
```

#### **object-name**

Input parameter that names a QMF procedure or query that will run after QMF starts. The parameter value, including the name and any substitution variable values that you pass, can be up to 32,704 bytes.

All types of QMF queries are accepted. The procedure can be either a QMF linear procedure or a procedure with logic.

The query or procedure that is named in this parameter must exist in the QMF catalog in the subsystem in which the stored procedure interface components are installed.

One result set is returned if the specified object is a query; up to 21 result sets can be returned from a procedure (including trace output returned as the last result set when *trace-level* is L2 and *L2-destination* is blank or null). QMF returns one result set to the calling program each time the following command is encountered in the procedure:

```
PRINT REPORT (PRINTER=' ')
```

Set the DSQEC\_CC global variable to 0 to eliminate carriage control characters from any returned result sets.

The PRINTER option can be set on the command itself or you can issue the following command in the procedure to set the PRINTER option of the QMF profile:

```
SET PROFILE (PRINTER=' ')
```

Because QMF cannot display panels when it has been started as a stored procedure, no prompting for variable values is available. If the object is a query, values for all variables in the query must be passed on the CALL statement. If the object is a procedure, the procedure can contain SET GLOBAL commands to set the necessary values before they are required. Values can also be passed on the CALL statement.

This parameter is ignored when *trace-level* is set to PTF.

### ***trace-level***

Input parameter that specifies the level of trace detail. Valid values include:

#### **blank**

No QMF trace output is generated.

#### **null**

No QMF trace output is generated.

#### **NONE**

No QMF trace output is generated.

#### **L2**

This option traces QMF messages and commands at the highest level of detail. Where the trace output is sent depends on how *L2-destination* is set.

#### **ALL**

This option traces QMF activity at the highest level of detail, including program initialization errors and other errors that might occur before the user's profile is established. Trace output is sent to the DSQDEBUG DD card that you defined when installing the QMF stored procedure interface.

#### **PTF**

This option is used to verify that the stored procedure interface is running correctly. Do not use this option unless instructed to do so in this information or by IBM Software Support.

You can include a SET PROFILE (TRACE command in the initial procedure that is specified by the *object-name* parameter to change the level of trace detail for the duration of the stored procedure session as long as trace output is set to go to the DSQDEBUG data set. If *trace-level* is set to L2, *L2-destination* must be set to DSQDEBUG for SET PROFILE (TRACE commands to be accepted.

### ***L2-destination***

Input parameter that specifies the destination for the trace log when *trace-level* is set to L2. Valid values include:

**DSQDEBUG**

Sends the trace output to the DSQDEBUG DD card that you defined when installing the QMF stored procedure interface. Use this option if the procedure passed in the *object-name* parameter is likely to contain SET PROFILE commands that include the TRACE option.

**blank**

Returns the trace output as the last result set from the stored procedure run.

**null**

Returns the trace output as the last result set from the stored procedure run.

The DSQSDBLG parameter, which appears in the DSQSCMDn exec, takes its value from this input parameter. This parameter cannot be externally set outside the context of the stored procedure interface.

**language**

Input parameter that specifies the language in which QMF will run.

Specify a 1-character national language identifier. A blank or null value for this input parameter invokes QMF in English.

**status-message**

Output parameter that contains the last message that was issued from the execution of the procedure or query that is passed in the *object-name* parameter. How the output parameter is defined depends on the software program that issues the CALL statement. For example, in QMF for Workstation, the output parameter containing the status message is defined as a question mark (?) character.

**Related concepts**Starting QMF as a Db2 for z/OS stored procedure

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

Getting the right level of detail in your trace output

You can trace all QMF functions in detail or trace individual QMF functions.

Tracing individual QMF modules

You can turn on a trace for certain modules using the SET PROFILE command and the module DSQUTRAC.

Viewing QMF trace data

DSQDEBUG holds the information that is recorded by the trace facility. It must be allocated before you start QMF if you want to use tracing.

**Related tasks**Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

Determining the QMF service level

Running an SMP/E report against the target or distribution zones is always the best way to determine the service level. However, you can also determine the QMF service level in other ways.

**Related reference**Examining error log reports

The Q.ERROR\_LOG table is a QMF control table that logs information about resource and other problems.

### Starting QMF for TSO from QMF for Workstation and running a linear procedure

This example shows how to start QMF for TSO from QMF for Workstation and run a simple linear procedure that queries the Q.STAFF sample table for rows that meet specific criteria for employee name, department, and job.

#### About this task

The query is named STAFFQUERY:

```
SELECT * FROM Q.STAFF
WHERE
NAME = &NAME AND
DEPT = &DEPT AND
JOB = &JOB
```

The example QMF procedure that runs this query is named STAFFPROC:

```
SET GLOBAL (DSQEC_CC=0 --Turn off carriage control for printing
RUN QUERY STAFFQUERY (&&NAME = &NAME &&DEPT = &DEPT &&JOB = &JOB
PRINT REPORT (PR= ' ')
```

Code 1 PRINT REPORT command for each result set to be returned to the calling program. To receive report output back in a result set, the PRINTER option of the PRINT REPORT command must be set to a string of blanks. If you have several PRINT REPORT commands in the procedure, you can issue a SET PROFILE command to set the PRINTER option before you code the first PRINT REPORT command so that you do not have to specify the PRINTER option multiple times.

The procedure can include up to 20 PRINT REPORT commands that return result sets. To eliminate carriage control characters from the result sets, set the DSQEC\_CC global variable to 0, as shown in the preceding example.

#### Procedure

- To start QMF for TSO as a Db2 for z/OS stored procedure and run the STAFFPROC example procedure, issue a command like one of the following from QMF for Workstation:
  - You can use parentheses as delimiters for the variable values:

```
CALL Q.DSQQMFSP('STAFFPROC(&NAME=(' PERNAL ' '),&DEPT=(20),&JOB=(' SALES ' '))','L2',' ','E',?)
```

This CALL statement:

- Returns to QMF for Workstation a result set containing the following row from the Q.STAFF sample table:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
20	PERNAL	20	SALES	8	18171.25	612.45

- Specifies a value of L2 for the *trace-level* parameter, which traces messages and commands.
  - Leaves the *L2-destination* parameter blank, which specifies that QMF returns the trace output as the last result set. Thus, a total of two result sets are returned when the STAFFPROC procedure ends.
  - Specifies English as the language in which QMF runs, denoted by a value of E for the *language* parameter.
  - How the output parameter is defined depends on the software program that issues the CALL statement. For example, in QMF for Workstation, the output parameter is defined as a question mark (?) character.
- You can also pass the variable values without using parentheses as delimiters:



```
CALL Q.DSQQMFSP('STAFFPROC(&NAME='peina1',&DEPT=20,&JOB='SALES'),'L2','','E',?)
```

- The following example CALL statement passes a variable value that contains an apostrophe:

```
CALL Q.DSQQMFSP('STAFFPROC(&NAME='O''BRIEN',&DEPT=38,&JOB='SALES'),'L2','','E',?)
```

This statement returns the following row to QMF for Workstation as the first result set:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
40	O'BRIEN	38	SALES	6	18006.00	846.55

The last result set contains any trace output from the stored procedure run.

### Starting QMF for TSO from QMF for Workstation and running a procedure with logic

In this example, you start QMF for TSO as a stored procedure from QMF for Workstation. You then run a procedure with logic that sets default values, retrieves global variables values, and runs a query.

#### About this task

The procedure with logic STAFFPROC, shown here, completes the following tasks:

- Makes the SHARE=YES parameter the default on all QMF SAVE commands.
- Sets the DSQEC\_CC global variable to 0 to eliminate carriage control characters from result sets that are returned to the calling program.
- Retrieves the value of the DSQAO\_STO\_PROC\_INT global variable, which indicates whether QMF for TSO was started as a Db2 for z/OS stored procedure.
- Writes message text to the trace log that indicates whether QMF for TSO was started as a Db2 for z/OS stored procedure.
- Runs a query named STAFFQUERY, which is as follows:

```
SELECT * FROM Q.STAFF
WHERE
NAME = &NAME AND
DEPT = &DEPT AND
JOB = &JOB
```

- Returns the query results to the calling program as a result set. For each result set that must be returned, code 1 PRINT REPORT command as follows:

```
PRINT REPORT (PRINTER=' ')
```

If you have several PRINT REPORT commands in the procedure, you can issue a SET PROFILE command, as shown in the following example, to set the PRINTER option of the profile to a string of blanks so that you do not have to specify this option on each PRINT REPORT command. The procedure can include up to 20 PRINT REPORT commands that return result sets.

```
/* REXX */
"SET GLOBAL (DSQEC_SHARE = 1"
"SET GLOBAL (DSQEC_CC=0"
"GET GLOBAL ("SPINT"=DSQAO_STO_PROC_INT)"
IF SPINT = 1 THEN
DO
"MESSAGE (TEXT 'YOU ARE RUNNING IN THE QMF STORED PROC INT.'"
"SET PROFILE (PR = ' ')"
END
ELSE
DO
"MESSAGE (TEXT 'YOU ARE NOT RUNNING IN THE QMF STORED PROC INT.'"
END

"RUN QUERY STAFFQUERY (&&NAME = &NAME &&DEPT = &DEPT &&JOB = &JOB"
"PRINT REPORT"
```

Figure 17. Procedure with logic, named STAFFPROC

## Procedure

To start QMF for TSO from QMF for Workstation and run a procedure with logic, follow this example:

1. Issue a command like the following from QMF for Workstation.

This command starts QMF for TSO as a Db2 for z/OS stored procedure and run the example procedure with logic:

```
CALL Q.DSQQMFSP('STAFFPROC(&NAME=' 'O' ' 'BRIEN' ',&DEPT=38,&JOB=' 'SALES' '),' 'L2' ',',' 'E' ',?)
```

This CALL statement:

- Uses double quotation marks as delimiters for the variable values.
- Returns to QMF for Workstation a result set containing the following row from the Q.STAFF sample table:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
40	O'BRIEN	38	SALES	6	18006.00	846.55

- Specifies a value of L2 for the **trace-level** parameter, which traces messages and commands.
- Leaves the **L2-destination** parameter blank, which specifies that QMF returns the trace output as the last result set. Thus, a total of two result sets are returned when the STAFFPROC procedure ends.
- Specifies English as the language in which QMF runs, denoted by a value of E for the **language** parameter.

How the output parameter is defined depends on the software program that issues the CALL statement.

For example, in QMF for Workstation, the output parameter is defined as a question mark (?) character.

2. To pass the entire text of a query when you start QMF, create a query that contains only a variable. For example:

```
&QUERY
```

3. Create a procedure, which is named RUNQPROC, that runs the query, as shown in the following example.

The same results can be achieved with a linear procedure.

```
/* REXX */  
"SET PROFILE (PR = ' ' "  
"RUN QUERY RUNQ(&&QUERY=&QUERY"  
"PRINT REPORT"
```

The following example CALL statement starts QMF for Workstation and runs the RUNQPROC procedure, passing the entire query text as part of the first parameter:

```
CALL Q.DSQQMFSP('RUNQPROC(&QUERY=((SELECT CURRENT SERVER FROM  
SYSIBM.SYSDUMMY1)))','L2',' ','E',?)
```

## Starting QMF on ISPF

You can let users start QMF using ISPF services.

You can add JCL to the ISPF environment that defines QMF resources. You can:

- Add QMF to an initial dialog of ISPF.
- Replace the initial dialog with one that starts QMF directly.
- Create a CLIST to start QMF as a program dialog.

You can also use any of the previous methods to start QMF using one of the other methods. For example, you can run an initial dialog from a CLIST.

If you use JCL that points to the QMF program location, the JCL must always be in an initial dialog.

To run QMF under ISPF, you must start the QMF program dialog using the ISPF SELECT service. When a TSO CALL statement or a TSO command is used, the results can be unpredictable.

### Restriction:

- You cannot run QMF as a command dialog. For example, the following statements are not valid:

```
ISPSEXEC SELECT CMD(DSQMF) NEWAPPL(DSQE)
ISPSTART CMD(DSQMF) NEWAPPL(DSQE)
```

- If QMF is started as an initial dialog, you cannot enter QMF from a split screen or create a split screen during a QMF session.

## Starting QMF from an ISPF menu

You can configure a menu option to start QMF.

This sample definition for the ISPF Master Application Menu shows how to add an option to the menu. In this definition, Option 2 was added for reaching QMF through a CLIST.

```
)BODY
%----- MASTER APPLICATION MENU -----
%SELECT APPLICATION ==>_OPT +
%                               +USERID -
%                               +TIME -
% 1 +SPF - SPF PROGRAM DEVELOPMENT FACILITY +TERMINAL -
% 2 +QMF - RUN QMF UNDER THE DB2T SUBSYSTEM +PF KEYS -%
%
%
%
%
%
%
%
%
%
%
%
% P +PARMS - SPECIFY TERMINAL PARAMETERS AND LIST/LOG DEFAULTS
% X +EXIT - TERMINATE USING LIST/LOG DEFAULTS
%
+PRESS%END KEY+TO TERMINATE +
%
)INIT
)PROC

&SEL = TRANS( TRUNC (&OPT, '.')
              1, 'PANEL(ISPOPRIM) NEWAPPL '
              2, 'PGM(DSQMF) NEWAPPL(DSQE) PASSLIB PARM(DSQSSUBS=DB2T) '
              /* ADD OTHER APPLICATIONS HERE */
              /*
              P, 'PANEL(ISPOPT) '
              X, 'EXIT'
              ', ', ', '
              ', ', ', '
              *, '? ' )
)END
```

Figure 18. Sample Master Application Menu

You can add more than one option to your menu. Suppose, for example, that DB2T is a test Db2 subsystem and DB2P is the production subsystem. In this case, you must add two options to your menu: one for each subsystem. You might have each option call a different CLIST, or you might create one CLIST with a positional parameter for the subsystem. The lines in the menu's PROC section might look like this:

```
2, 'PGM(DSQMF) NEWAPPL(DSQE) PASSLIB PARM(DSQSSUBS=DB2T) '
3, 'PGM(DSQMF) NEWAPPL(DSQE) PASSLIB PARM(DSQSSUBS=DB2P) '
```

## Using LIBDEF statements to allocate QMF program libraries

If you want to use ISPF LIBDEF statements to allocate QMF libraries during an ISPF session, allocate the program libraries to a unique QMF ddname of DSQLLIB.

Specify ddname DSQLLIB as the ID value on the LIBRARY option of the ISPF LIBDEF statement.

For example, to allocate QMF and Db2 product libraries, write a TSO ALLOCATE and ISPF LIBDEF statement:

```
ALLOC FI(DSQLLIB) DA('QMF1210.SDSQEXIT','QMF1210.SDSQLOAD', +  
'DSN1110.SDSNEXIT','DSN1110.SDSNLOAD') SHR REUSE  
LIBDEF ISPLLIB LIBRARY ID(DSQLLIB)
```

The GDDM external defaults module, ADMADFT, is not found in the search order that is defined by the LIBDEF statement for DSQLLIB. Sample allocation statements for GDDM modules are described in the information about preparing the TSO logon procedure.

To allocate program libraries that use the ISPF LIBDEF service, write a CLIST similar to the one shown here. The following CLIST assumes that ISPF is already running and has other ISPF resources already allocated:

```

/*****
/* Allocate QMF and DB2 programs to DSQLLIB */
/*****
ALLOC FI(DSQLLIB) SHR REUSE +
      DA('QMF1210.SDSQEXIT', +
        'QMF1210.SDSLOAD', +
        'DSN1110.SDSNEXIT', +
        'DSN1110.SDSNLOAD')
/*****
/* Allocate QMF libraries used for GDDM */
/*****
ALLOC FI(ADMGGMAP) DA('QMF1210.SDSQMAPE') SHR REUSE
ALLOC FI(ADMCFORM) DA('QMF1210.SDSQCHRT') SHR REUSE
ALLOC FI(DSQUCFRM) DA('QMF1210.DSQUCFRM') SHR REUSE
ALLOC FI(ADMGDF) DA('QMF1210.CHARTLIB') SHR REUSE
/*****
/* Allocate QMF product data sets */
/*****
ALLOC FI(DSQPRINT) SYSOUT(Z) LRECL(133) RECFM(F B A ) BLKSIZE(1330)
ALLOC FI(DSQPNLE) DA('QMF1210.DSQPNLE') SHR REUSE
ALLOC FI(DSQDEBUG) SYSOUT(Z) LRECL(121) RECFM(F B A) BLKSIZE(1210)
ALLOC FI(DSQDUMP) SYSOUT(Z) LRECL(125) RECFM(V B A) BLKSIZE(1632)
ALLOC FI(DSQSPILL) NEW UNIT(SYSDA) SPACE(10,20) CYLINDERS
ALLOC FI(DSQEDIT) NEW UNIT(SYSDA)
/*****
/* Issue ISPF LIBDEF for QMF libraries used for ISPF */
/*****
ISPEXEC LIBDEF ISPLLIB LIBRARY ID(DSQLLIB)
ISPFEXE LIBDEF ISPLLIB DATASET ID('QMF1210.SDSQPLBE')
ISPFEXE LIBDEF ISPLLIB DATASET ID('QMF1210.SDSQSLBE')
ISPFEXE LIBDEF ISPLMLIB DATASET ID('QMF1210.SDSQMLBE')
ISPEXEC LIBDEF ISPTLIB DATASET ID('QMF1210.SDSQTLBE')
/*****
/* Start QMF dialog using PASSLIB */
/*****
ISPEXEC SELECT PGM(DSQQMF) NEWAPPL(DSQE) PASSLIB
/*****
/* Free ISPF LIBDEF for QMF libraries used for ISPF */
/*****
ISPEXEC LIBDEF ISPLLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPLLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPLLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPLMLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPTLIB LIBRARY ID( )
FREE FI(DSQLLIB)
/*****
/* Free QMF product data sets */
/*****
FREE FI(DSQPRINT)
FREE FI(DSQPNLE)
FREE FI(DSQDEBUG)
FREE FI(DSQDUMP)
FREE FI(DSQSPILL)
FREE FI(DSQEDIT)
/*****
/* Free QMF libraries used for GDDM */
/*****
FREE FI(ADMGGMAP)
FREE FI(ADMCFORM)
FREE FI(DSQUCFRM)
FREE FI(ADMGDF)

```

Figure 19. ISPF LIBDEF CLIST

This CLIST shows a file allocation for spill data (DSQSPILL). You can spill data to extended storage if you do not want to allocate a file for spill data.

### Related tasks

[Preparing the TSO logon procedure](#)

You must modify the TSO logon procedure to support the storage and other requirements of your users. This step includes allocating load libraries and data sets.

[Spilling report data to extended virtual storage \(TSO only\)](#)

In QMF for TSO, use extended storage for spill data unless the system on which QMF is running has very limited extended storage available.

## Starting QMF in batch mode in ISPF

You can start QMF in batch mode to potentially save resources and time.

### Procedure

You can start QMF using ISPF with or without using a CLIST.

- Place either of the following statements in the SYSTSIN data set of your JCL:

- Without a CLIST

```
ISPSTART PGM(DSQMF) NEWAPPL(DSQE) PARM(. . .DSQSMODE=B,DSQSRUN=auth_ID.proc_name)
```

The parameters following the PARM keyword establish batch mode (DSQSMODE=B), identify the procedure to be run (DSQSRUN=*auth\_ID.proc\_name*), and can include variables for that procedure.

The ellipsis after the PARM keyword represents optional parameter values that the user might want to include in addition to the required DSQSMODE and DSQSRUN parameters.

The value you specify for the DSQSRUN parameter must contain the authorization ID of the owner. For example, the following statement specifies a procedure named PROCA, which is owned by user JONES:

```
ISPSTART PGM(DSQMF) NEWAPPL(DSQE) PARM(DSQSMODE=B,DSQSRUN=JONES.PROCA)
```

After the procedure runs, QMF ends and returns control to ISPF. ISPF can then continue with another procedure or command. When ISPF stops, TSO runs the next TSO command in SYSTSIN. When all commands in the SYSTSIN data set have been run, the job step ends.

- With a CLIST

```
ISPSTART CMD(clist_name) NEWAPPL
```

where *clist\_name* is the name of the CLIST that starts QMF.

## Examples of starting QMF under ISPF

These examples show how to start and pass parameters to QMF under ISPF.

You might want to start QMF under ISPF in these circumstances

- Starting ISPF from a CLIST and specifying QMF as the initial dialog:

```
ISPSTART PGM(DSQMF) NEWAPPL(DSQE) PARM(DSQSIROW=150)
```

This statement passes a value of 150 for DSQSIROW (which controls the number of rows fetched before the first screen of the report is displayed).

- Starting from a CLIST that operates within ISPF:

```
ISPEXEC SELECT PGM(DSQMF) NEWAPPL(DSQE) PARM(DSQSSUBS=SFDX)
```

This statement passes the name SFDX for the local Db2 for z/OS subsystem.

- Starting from an ISPF menu:

```
)PROC  
&SEL = TRANS( TRUNC (&OPT, '.' )  
              1, 'PGM(DSQMF) NEWAPPL(DSQE) PARM(DSQSPILL=NO) '  
              .  
              .  
              .
```

This code passes a value of NO for the DSQSPILL parameter, which indicates to not use spill storage for report data.

- Starting from a CLIST and specifying an initial procedure:

```
ISPSTART PGM(DSQMF) NEWAPPL(DSQE)
PARM(DSQSRUN=Q.IPROC(&&&TABLE=Q.STAFF))
```

This statement uses the DSQSRUN parameter to:

- Specify an initial procedure, Q.IPROC, to run when QMF starts
- Pass a value, Q.STAFF, to the procedure for the variable &TABLE

The DSQSRUN parameter as specified in this example results in the following QMF command:

```
RUN Q.IPROC(&TABLE=Q.STAFF
```

You must precede the variable with the correct number of ampersands in the initial procedure.

### Related concepts

[Passing variable values to an initial procedure](#)

When you supply the name of an initial procedure on the DSQSRUN parameter, you can also supply values for one or more variables contained in the procedure.

## Starting QMF in native z/OS batch

---

You can allow your users to start QMF in native z/OS as a batch job

Create JCL that defines where the panels are stored, the file names of the panels, the names and locations of other tables and QMF objects, and where the spill file is located (if QMF is configured to spill data to a file instead of to extended storage).

To issue QMF commands in native z/OS batch, use the following QMF program parameters:

- DSQSMODE

Use a value of B for batch mode. (I specifies interactive operation.) The short name of this parameter is M. This sample command sets the mode to batch by specifying M=B on the invocation command.

- DSQSRUN

Use this parameter to specify the name of an initial procedure to run when QMF starts. The short name of this parameter is I. This sample command that specifies a procedure named X on the I parameter.

In the following example, QMF terminates when procedure X completes. You can check the return code of the job to determine if the job ran successfully.

```

//RUNQMF EXEC PGM=DSQMF, PARM='M=B, I=X, P=QMF12, S=DSN'
//*****
//*      Program load libraries
//*****
//STEPLIB DD DSN=QMF1210.SDSQLOAD, DISP=SHR
//          DD DSN=DSN1110.SDSNEXIT, DISP=SHR
//          DD DSN=DSN1110.SDSNLOAD, DISP=SHR
//          DD DSN=GDDM.SADMMOD, DISP=SHR
//*****
//*      QMF/GDDM maps
//*****
//ADMGGMAP DD DSN=QMF1210.SDSQMAPE, DISP=SHR
//*****
//*      Data sets used by QMF      *
//*****
//DSQPRINT DD SYSOUT=*, DCB=(RECFM=FBA, LRECL=133, BLKSIZE=1330)
//DSQDEBUG DD SYSOUT=*, DCB=(RECFM=FBA, LRECL=122, BLKSIZE=1210)
//DSQDUMP DD SYSOUT=*, DCB=(RECFM=VBA, LRECL=125, BLKSIZE=1632)
//DSQPILL DD DSN=&&SPILL, DISP=(NEW, DELETE),
// UNIT=SYSDA, SPACE=(TRK, (100), RLSE),
// DCB=(RECFM=F, LRECL=4096, BLKSIZE=4096)

```

Figure 20. JCL to run a QMF procedure in native z/OS batch

If you are running QMF in native z/OS, data set names that are used in QMF procedures must be fully qualified. The TSO prefix and suffix are not available in native z/OS.

### Related concepts

[Setting program parameters and preferences at startup time](#)

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

## Starting QMF on CICS

After QMF is customized for CICS, you can start QMF from a CICS display.

To start QMF on CICS, use the transaction name QMF*n*. *n* is the 1-character National Language Feature identifier that represents the language in which QMF will run. Alternatively, you can write an application that uses the CICS START command. The following command shows the default (English) transaction on QMF for CICS.

```
QMF parameters
```

QMF is the transaction ID, and *parameters* represents the program parameters that you want.

You can also write an application program to issue the CICS START command and specify program parameters, as in the following example:

```
EXEC CICS START TRANSID(QMFE) FROM (parameters) TERMID('id')
```

A terminal ID (TERMID) is required for an interactive session (DSQSMODE=I), and is optional for a noninteractive session (DSQSMODE=B). If the terminal ID specifies where the calling CICS application is running, the QMF session starts when the CICS application finishes. To specify a terminal ID, the device that is described by the ID must be available. Also, make sure that the ID is defined as either local or remote on the system where the START command is issued.

The following examples show starting QMF under CICS:

- Starting from a cleared CICS screen:

```
QMFE DSQSIROW=150, DSQSBSTG=500000, DSQSPILL=NO
```



This statement passes a value of 150 for DSQSIROW (rows fetched before the first screen of the report is displayed). The statement also passes a value of 500,000 bytes for DSQSBSTG (maximum storage for reports), and specifies no QMF spill file (DSQSPILL=NO).

- Starting from a cleared CICS screen and specifying an initial procedure:

```
QMFE DSQSRUN=Q.IPROC(&&TABLE=Q.STAFF)
```

This statement uses the DSQSRUN parameter to:

- Specify an initial procedure, Q.IPROC, to run when QMF starts
- Pass a value (Q.STAFF) to the procedure for the variable &TABLE

The DSQSRUN parameter as specified in this example results in the following QMF command:

```
RUN Q.IPROC(&TABLE=Q.STAFF)
```

You must precede the variable with the correct number of ampersands in the initial procedure.

### Related concepts

#### Passing variable values to an initial procedure

When you supply the name of an initial procedure on the DSQSRUN parameter, you can also supply values for one or more variables contained in the procedure.

#### Setting program parameters and preferences at startup time

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Initializing global variables and QMF session behavior when QMF starts

You can use several methods to set QMF global variables when QMF starts. These methods involve modifying the DSQUOPTS routine, the Q.GLOBAL\_VARS global variable table, or the default system initialization procedure. This procedure can also be used to set other aspects of a user's QMF session before the home panel displays.

When QMF starts, it checks the following sources of global variable settings in order, starting with the default values that are defined by QMF. If a global variable is defined in multiple places, QMF uses the last value that it finds.

1. Default global variable values that are defined by QMF.
2. Global variable values that are defined in the DSQUOPTS exit routine.
3. Global variable values that are defined in the Q.GLOBAL\_VARS global variable table with a USERID of SYSTEM. These are global variable values that are set by an administrator.
4. Global variable values that are defined in the Q.GLOBAL\_VARS global variable table with a USERID of a user. These are global variable values that are saved from a previous session.
5. Global variable values that are defined by the default system initialization procedure (Q.SYSTEM\_INI).

**Restriction:** If a global variable is defined in the Q.GLOBAL\_VARS table with a value of 0 in the UPDATEABLE column, that global variable cannot be set by the Q.SYSTEM\_INI procedure. The Q.SYSTEM\_INI procedure also cannot update global variables that are defined by QMF as read-only, such as those that begin with DSQAO.

## Setting global variables with the DSQUOPTS routine

The settings in the DSQUOPTS exit routine override the values of certain global variables when QMF starts. You can modify the default DSQUOPTS routine to suit your needs.

### About this task

The default version of DSQUOPTS is shipped in the QMF1210.SDSQLOAD data set.

### Procedure

To modify the DSQUOPTS exit routine, follow these steps:

1. Change the Assembler source as necessary. The source for the exit resides in QMF1210.SDSQSURE(DSQUOPTS). For details on how to specify override values, see the DSQUOPTS prolog.
2. Assemble and link-edit the module. A sample job that assembles and link-edits DSQUOPTS resides in member QMF1210.SDSQSAPE(DSQ1UOPT).
3. Allocate the QMF exit library in the initialization procedure you are using so that the modified exit is referenced during the QMF initialization process.

### Results

When you use the sample job to assemble and link-edit the DSQUOPTS exit routine, the modified DSQUOPTS load module is placed in the default QMF exit library, QMF1210.SDSQEXIT.

### Variables that can be set in the DSQUOPTS routine

The DSQUOPTS exit routine provides settings for some variables. You can use the DSQUOPTS exit to set other variable values at initialization time as well.

The default version of the DSQUOPTS exit routine sets these variable values:

#### DISADM

Determines whether QMF checks whether user IDs have QMF administrator authority. You can also use QMF global variable DSQEC\_DISABLEADM to set this value.

Valid values are:

##### NO

Enables checking for QMF administrator authority. This value is the default.

When the DISADM variable is set to NO, the DSQEC\_DISABLEADM global variable has a value of 0.

##### YES

Disables checking for QMF administrator authority.

When the DISADM variable is set to YES, the DSQEC\_DISABLEADM global variable has a value of 1.

During QMF initialization, QMF checks whether the authorization ID of the user who is starting QMF has either the INSERT or DELETE privilege on the Q.PROFILES table. If the authorization ID has either of these privileges, QMF considers the user to be a QMF administrator and sets the global variable DSQAO\_QMFADM to 1 to reflect this authority. QMF administrators can perform the following operations on QMF objects that they do not own, without requiring those objects to be shared with all QMF users:

- SAVE
- ERASE
- IMPORT
- EXPORT
- DISPLAY

QMF administrators can also issue the DISPLAY command on ANALYTIC objects and the ERASE command on ANALYTIC and FOLDER objects that are owned by other users.

If during this checking at initialization time the user who started QMF is not found to have QMF administrator authority, a -551 SQL code is issued for the INSERT privilege, the DELETE privilege, or both. This SQL code is expected under these circumstances. The message is not displayed to the user, nor is it captured in the QMF trace data set (DSQDEBUD) or in the error log table (Q.ERROR\_LOG), because it is not an error. The message does not stop QMF initialization or cause any problems during QMF execution. However, to suppress this SQL code for any reason, you must disable administrator authority checking.

If you disable QMF administrator authority checking, you might be limiting your ability to work freely with other users' objects. Without QMF administrator authority, every object that you work with must be shared with all users. To share objects, you can either request that users save all objects with the SHARE=YES parameter or you can set the DSQEC\_SHARE variable to change the default value for the SHARE parameter to YES. This value shares all objects with all users so that users must then explicitly specify SHARE=NO when they save objects that they do not want to share. Setting the DSQEC\_SHARE variable prevents you from having to take extra time to switch user IDs or share objects when you need to provide user assistance.

## SHARE

Sets the default value for the SHARE parameter of the SAVE command and the Save function key in QMF Analytics for TSO. You can also use QMF global variable DSQEC\_SHARE to set this value.

Valid values are:

### NO

Makes NO the default setting for the SHARE parameter of the QMF SAVE command. This value is the default.

When the SHARE variable is set to NO, the DSQEC\_SHARE global variable has a value of 0.

### YES

Makes YES the default setting for the SHARE parameter.

When the SHARE variable is set to YES, the DSQEC\_SHARE global variable has a value of 1.

## LASTRUN

Determines the command or set of commands that causes the LAST\_USED column of the Q.OBJECT\_DIRECTORY table to be updated. This value is displayed in the **Last Used** field on object lists that are generated by the LIST command. You can also use QMF global variable DSQEC\_LAST\_RUN to set this value. The value in the LAST\_USED column is updated regardless of whether the issued command is successful. However, in some cases, the LAST\_USED column is not updated immediately, and if QMF is terminated abnormally, the column might not be updated.

Valid values are:

### ZERO

Causes the LAST\_USED column of the Q.OBJECT\_DIRECTORY table to be updated when any of the following commands is run:

- CONVERT
- DISPLAY
- EXPORT
- IMPORT
- LAYOUT
- PRINT
- RUN
- SAVE

This value is the default.

When the LASTRUN variable is set to ZERO, the DSQEC\_LAST\_RUN global variable has a value of 0.

**ONE**

Restricts updates of the LAST\_USED column to RUN, SAVE, and IMPORT commands only.

When the LASTRUN variable is set to ONE, the DSQEC\_LAST\_RUN global variable has a value of 1.

**TWO**

Restricts updates of the LAST\_USED column to RUN commands only.

When the LASTRUN variable is set to TWO, the DSQEC\_LAST\_RUN global variable has a value of 2.

**OTC\_LICENSE**

This variable is obsolete and is included only for compatibility. The only valid value is NOT\_USED.

**Related concepts**

Required authorities for QMF administration

To avoid issuing multiple GRANT statements for specific database objects, you must use an authorization ID with DBADM or equivalent authority for most QMF administration and customization tasks. You need this level of authority on any database in which your users store data.

**Setting global variables with the global variable table**

You can use the Q.GLOBAL\_VARS table to set initial global variable values that are to be used when QMF starts. You can also specify whether those global variables can be changed during the session.

**About this task**

You can set initial values for both user-defined global variables and QMF predefined global variables that are not read-only.

**Procedure**

To set initial global variable values in the Q.GLOBAL\_VARS table:

1. Write SQL INSERT statements to add rows to the Q.GLOBAL\_VARS table. Specify values for all columns as follows:

**VARNAME**

Specify the name of the global variable.

**UPDATEABLE**

Specify a value of 0 to determine that the global variable cannot be changed by users after QMF initialization. Specify a value of 1 to determine that the global variable can be updated during the QMF session.

**USERID**

By default, a value of SYSTEM is used and does not need to be specified.

**VARVALUE**

Specify the value of the global variable.

2. Run the INSERT statements.

**Example**

See the following example statements for inserting rows into the Q.GLOBAL\_VARS table.

<i>Table 30. Sample INSERT statements to add rows to the Q.GLOBAL_VARS table</i>	
<b>INSERT statement</b>	<b>Description</b>
INSERT INTO Q.GLOBAL_VARS (VARNAME,VARVALUE,UPDATEABLE) VALUES('DSQDC_SCROLL_AMT','CSR',1)	Sets the DSQDC_SCROLL_AMT global variable to CSR and specifies that users can change the global variable.
INSERT INTO Q.GLOBAL_VARS (VARNAME,VARVALUE,UPDATEABLE) VALUES('DSQEC_DISABLEADM','0',0)	Sets the DSQEC_DISABLEADM global variable to 0 and specifies that users cannot change the global variable.

Table 30. Sample INSERT statements to add rows to the Q.GLOBAL\_VARS table (continued)

INSERT statement	Description
INSERT INTO Q.GLOBAL_VARS (VARNAME,VARVALUE,UPDATEABLE) VALUES('MYVAR1','VALUE OF MYVAR1',1)	Sets the MYVAR1 global variable to VALUE OF MYVAR1 and specifies that users cannot change the global variable.
INSERT INTO Q.GLOBAL_VARS (VARNAME,VARVALUE,UPDATEABLE) VALUES('MYVAR2','VALUE OF MYVAR2',2)	Sets the MYVAR2 global variable to VALUE OF MYVAR2 and specifies that users cannot change the global variable.

#### Structure of the Q.GLOBAL\_VARS table

The Q.GLOBAL\_VARS table stores global variable values and session variable values. This table is created at QMF installation and is required for QMF to start, except on DB2 for VSE and VM.

A QMF administrator can use the Q.GLOBAL\_VARS table to specify global variable values that are to be set when QMF starts. Depending on the value of the UPDATEABLE column for each defined global variable, users can change these global variable values after QMF initialization. By default, a global variable value is retained until a user resets it or ends the QMF session. However, the DSQEC\_USERGLV\_SAV global variable can be set to save global variable values from one session to another. In this case, a user's global variable values are also stored in this table.

The DSQEC\_SESSGLV\_SAV global variable can also be set to save a user's input on QMF panels, both within a session and across sessions. In this case, the user's panel input is saved in the Q.GLOBAL\_VARS table as session variables, which have a prefix of DXY.

Although the table can be empty, columns cannot contain null values.

Table 31. Structure of the Q.GLOBAL\_VARS table

Column name	Data type and length	Function and possible values
VARNAME	CHAR(18)	<p><b>Function</b> Specifies the name of the global variable or session variable.</p> <p><b>Values</b> Any valid global variable name or session variable name.</p>

Table 31. Structure of the Q.GLOBAL\_VARS table (continued)

Column name	Data type and length	Function and possible values
UPDATEABLE	CHAR(1)	<p><b>Function</b> Specifies whether the global variable value can be changed by users after QMF initialization.</p> <p><b>Values</b></p> <p><b>0</b> Specifies that the global variable cannot be updated during the QMF session.</p> <p><b>1</b> Specifies that the global variable can be updated during the QMF session.</p> <p>Some QMF global variables, such as those that record information about the state of the current QMF session, are read-only. If you include a read-only global variable in the Q.GLOBAL_VARS table, the variable is not set at initialization, regardless of the UPDATEABLE setting. QMF initialization also fails.</p> <p>For global variables that are saved across sessions, this value is set according to whether the variable can currently be updated by the user. If the variable cannot be updated, this value is set to 0. Otherwise, it is set to 1.</p>
USERID	VARCHAR(128)	<p><b>Function</b> Specifies a QMF user ID.</p> <p><b>Values</b></p> <p><b>SYSTEM</b> This value must be used when initial global variable values are added to the table by an administrator. This value specifies that the global variable applies to all users. SYSTEM is the default value.</p> <p><b>userid</b> When global variable and session values are added to the table, this value is set to the user's QMF ID.</p> <p>The USERID and VARNAME fields are used as the primary key for each global variable entry.</p>
VARVALUE	VARCHAR(2000) At installation, the VARVALUE column is defined with a length of 2000. However, any length from 18 to 32555 can be used.	<p><b>Function</b> Specifies the value of the global variable or session variable.</p> <p><b>Values</b> Any valid global variable value.</p>

## Initialization with the default system initialization procedure

The name of the default procedure in QMF is Q.SYSTEM\_INI. The Q.SYSTEM\_INI procedure can run any QMF command or any stored query that the user is authorized to run before the home panel displays.

### About this task

The Q.SYSTEM\_INI procedure runs just after QMF initialization completes and before the QMF initial procedure specified by the DSQSRUN program parameter is run. Thus, if you have session controls in the procedure that is specified by the DSQSRUN parameter, consider moving them to the Q.SYSTEM\_INI procedure. All of the QMF functions available to QMF procedures are also available for use by the Q.SYSTEM\_INI procedure.

This sample Q.SYSTEM\_INI procedure is provided with QMF and makes SHARE=YES the default for all QMF objects that are saved in the database:

```
-- Q M F      S Y S T E M      I N I T I A L I Z A T I O N      P R O C
-- -----
--
-- FUNCTION:  PROVIDE AN EXAMPLE QMF SYSTEM INITIALIZATION PROCEDURE
--            THAT CAN BE ADDED AFTER QMF INSTALLATION. YOU MAY MOD-
--            IFY OR REPLACE THIS PROCEDURE WITH YOUR OWN VERSION.
--
--            THE PROCEDURE MUST BE STORED IN THE DATABASE UNDER THE
--            NAME OF Q.SYSTEM_INI BEFORE IT WILL RUN AUTOMATICALLY.
--            -----
--
-- THE COMMAND BELOW IS AN EXAMPLE OF ESTABLISHING A NEW DEFAULT
-- FOR THE SHARE OPTION OF THE SAVE COMMAND THAT WILL APPLY TO ALL
-- QMF USERS. (REMOVE THE LEADING COMMENT SYMBOLS "--" TO ACTIVATE
-- IT.)
--
-- SET GLOBAL (DSQEC_SHARE=1  -- MAKE SHARE=YES THE DEFAULT FOR ALL
```

Figure 21. The Q.SYSTEM\_INI procedure that is shipped with QMF

The Q.SYSTEM\_INI procedure, as well as any object used or called by this procedure, has the same security as any other QMF object or database object during a QMF session. The Q.SYSTEM\_INI procedure is a normal QMF procedure, except for the fact that QMF tries to run it each time a QMF session is started. Thus, to run the procedure successfully, QMF must be started under an authorization ID that has QMF administrator authority.

If the Q.SYSTEM\_INI procedure exists but is restricted (not shared), the result is the same as with any other QMF procedure object. If the procedure does not exist, QMF does not attempt to run it.

### Procedure

To use the default system initialization procedure, complete these steps:

#### 1. Install the procedure.

The sample procedure that is shipped with QMF is called DSQ0BINI. It can be found in QMF1210.SDSQSAPE(DSQ0BINI).

- If you installed QMF Version 12.1 into a database that contains a prior QMF release, issue the following command from within QMF to check for an existing system initialization procedure before you install the sample. Be sure that you are connected to the database under a QMF user ID with administrator authority.

```
DISPLAY Q.SYSTEM_INI
```

- If you already have a system initialization procedure and want to overwrite it with the sample, or do not have one and want to install the sample, continue with the following command:

```
IMPORT PROC FROM 'QMF1210.SDSQSAPE(DSQ0BINI)'
```

2. Save the procedure under the name Q.SYSTEM\_INI in the database in which it will run.

Before the default system initialization procedure can run automatically, you must save the procedure with the name Q.SYSTEM\_INI. Share the procedure with all QMF users.

a) You can share the procedure by displaying it in QMF and then issuing the command `SAVE PROC (SHARE=YES`.

b) Add a comment that describes the procedure when you save it.

For example:

```
SAVE PROC AS Q.SYSTEM_INI (SHARE=YES,COMMENT='QMF System Initialization Procedure')
```

### What to do next

To troubleshoot problems with the Q.SYSTEM\_INI procedure, you can use the QMF L2 tracing option to see commands and messages issued.

The Q.SYSTEM\_INI procedure cannot be replaced by other procedures, but it can call other procedures. If the default system initialization procedure does not meet your site's needs, you can create your own version

### Related concepts

[Required authorities for installing and administering QMF](#)

Specific authorities are required for installing QMF and for general QMF administration.

### Related tasks

[The trace facility](#)

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

## Creating your own initialization procedure

You can create your own procedure if the default version does not meet your site's needs.

Your QMF session procedure can be as simple as setting some QMF global variables or profile values, or as complex as providing a customized interface to QMF. Use the Q.SYSTEM\_INI procedure to call your initialization procedure. The Q.SYSTEM\_INI procedure cannot be replaced by site-specific procedures, but can be used to call other procedures.

For examples of how to create your own system initialization procedure, see one of the following sections:

- [“Creating a procedure that sets user-specific defaults” on page 138](#)
- [“Creating a procedure that displays an object list” on page 139](#)

### Creating a procedure that sets user-specific defaults

The Q.SYSTEM\_INI procedure can call another procedure. The procedure being called can be a user procedure that is created, owned, and updated by a QMF user.

Each user runs under a unique SQL ID. That SQL ID is the default object owner when the object owner is not otherwise specified. Thus, you can use the same name for each user's system initialization procedure, but modify it according to each user's needs. The following example shows a system initialization procedure that sets global variables and then calls a user-specific session procedure. In the example, the user-specific session procedure is called USER\_INI.



```

PROC                Q.SYSTEM_INI                LINE 1
-- This QMF procedure example shows how to set up QMF session defaults for
-- every QMF user and then calls a user procedure named USER_INI that sets
-- individual QMF session defaults.
--
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1)           -- Process English commands
QMF RESET PROC                                  -- Hide contents of this procedure
QMF SET PROFILE (WIDTH=80,LENGTH=66)           -- Set default report page size
QMF SET PROFILE (SPACE='DATABASE "DSQDBDEF"') -- Set default database for SAVE DATA command
QMF SET GLOBAL (DSQDC_LIST_ORDER=5D)          -- Sort object list by date modified
QMF SET GLOBAL (DSQEC_RESET_RPT=1)            -- Prompt for report completion
RUN USER_INI                                    -- Run user's session procedure
QMF END                                        -- Display QMF home panel first
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)          -- Return to presiding language

```

Figure 22. Q.SYSTEM\_INI example that calls a user-defined procedure

This example of the USER\_INI procedure is referenced in the Q.SYSTEM\_INI example procedure.

```

PROC                WILLIAMS.USER_INI           LINE 1
-- This QMF procedure example shows how to set up QMF session defaults for
-- a QMF user. The following settings replace any settings set by the
-- SYSTEM_INI proc.
--
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1)           -- Process English commands
QMF RESET PROC                                  -- Hide contents of this procedure
QMF SET PROFILE (SPACE='DATABASE "UIDDB"')     -- Store data in my own database
QMF SET PROFILE (PRINTER=MYROOM)               -- Print reports on a printer named MYROOM
QMF SET GLOBAL (DSQDC_LIST_ORDER=3A)          -- Sort object list by object name
QMF SET GLOBAL (DSQEC_RESET_RPT=2)            -- Always reset reports
QMF SET GLOBAL (DSQEC_SHARE=1)                -- Always share this user's QMF objects
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)          -- Return to presiding language

```

Figure 23. Example of a user procedure that is called by the system initialization procedure

## Creating a procedure that displays an object list

The next example is a system initialization procedure that displays a list of objects instead of the QMF home panel:

```

PROC                Q.SYSTEM_INI                LINE 1
-- This QMF procedure example shows how to set up QMF session defaults for
-- every QMF user to display a list of objects instead of the QMF home
-- panel.
--
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1)           -- Process English commands
QMF RESET PROC                                  -- Hide contents of this procedure
QMF SET GLOBAL (DSQDC_LIST_ORDER=3A)          -- Sort object list by object name
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)          -- Return to presiding language
QMF LIST ALL                                    -- List all objects

```

Figure 24. Using Q.SYSTEM\_INI to display a list of objects rather than the QMF home panel

## Related tasks

### Customizing users' object lists

Using the default views supplied by QMF for your table lists and column information can increase processing time, because Db2 gathers authorization information from the SYSIBM.SYSTABAUTH table. If you do not need the extra security provided by these authorization checks, consider creating your own views that generate a list of objects stored in the database.



---

## Chapter 10. Setting program parameters and preferences at startup time

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

When QMF for TSO is started as a Db2 for z/OS stored procedure, you can set some these program parameters as described when in the installation procedure for the stored procedure interface.

### **Related tasks**

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

---

### Summary of program parameters

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

The following table shows the long and short names of the QMF program parameters and the environments in which they can be used. Parameters that are used only in CICS do not have a short name.

Table 32. Program parameters

Long name	Short name	Environment	Description
DSQSBSTG	B	TSO, CICS	<p><b>Controls or sets</b> Maximum amount of virtual storage per user that is to be used for report operations.</p> <p><b>Valid values</b></p> <p><b>0 - 2147483647</b> Specifies storage in bytes.</p> <p><b>OKB - 2097152KB</b> Specifies storage in KB.</p> <p><b>OMB - 2048MB</b> Specifies storage in MB.</p> <p><b>OGB - 2GB</b> Specifies storage in GB.</p> <p><b>1% - 100%</b> Specifies a percent of available storage. Percentages are valid in TSO only.</p> <p><b>Defaults</b></p> <p><b>0</b> When you start QMF in TSO, ISPF, or native z/OS.  If you set this parameter to a value of less than 15000 in these environments, 15000 is used automatically.</p> <p><b>500000</b> When QMF is started in CICS</p> <p>For more information, see <a href="#">“Defining a fixed amount of virtual storage for reports”</a> on page 151</p>
DSQDBCS	K	TSO, CICS	<p><b>Controls or sets</b> Printing of DBCS data from non-DBCS display devices</p> <p><b>Valid values</b> YES (supports printing DBCS data from non-DBCS display devices) or NO (does not set this support)</p> <p><b>Default</b> NO</p> <p>For more information, see <a href="#">“Printing DBCS data from non-DBCS display devices”</a> on page 169.</p>

Table 32. Program parameters (continued)

Long name	Short name	Environment	Description
DSQSDBNM	D	TSO, CICS	<p><b>Controls or sets</b> Name of initial database location to connect to before the QMF home panel is displayed</p> <p><b>Valid values</b> Any valid database name; location names can be up to 16 characters. Do not use this parameter when you start QMF for TSO as a Db2 for z/OS stored procedure.</p> <p><b>Default</b> For CICS, the default database that is currently being used by CICS. Otherwise, the the default database for the subsystem that is being used.</p> <p>For more information, see <a href="#">“Specifying the initial database connection”</a> on page 149.</p>
DSQSDBQN	—	CICS	<p><b>Controls or sets</b> Name of CICS resource to use for QMF trace data</p> <p><b>Valid values</b> Any name that follows CICS queue-naming conventions</p> <p><b>Default</b> DSQD</p> <p>For more information, see <a href="#">“Setting the trace for CICS”</a> on page 168.</p>
DSQSDBQT	—	CICS	<p><b>Controls or sets</b> Type of CICS resource to use for QMF trace data</p> <p><b>Valid values</b> TS (uses a CICS temporary storage queue) or TD (uses a CICS transient data queue)</p> <p><b>Default</b> TD</p> <p>For more information, see <a href="#">“Setting the trace for CICS”</a> on page 168.</p>
DSQSDEBUG	T	TSO, CICS	<p><b>Controls or sets</b> Level of trace detail</p> <p><b>Valid values</b> ALL (traces all functions at the highest level of detail) or NONE (no trace data is recorded). Do not use this parameter when you start QMF for TSO as a Db2 for z/OS stored procedure.</p> <p><b>Default</b> NONE</p> <p>For more information, see <a href="#">“Setting the trace for TSO”</a> on page 168 and <a href="#">“Setting the trace for CICS”</a> on page 168.</p>

Table 32. Program parameters (continued)

Long name	Short name	Environment	Description
DSQSFISO	ISO	TSO, CICS	<p><b>Controls or sets</b> Controls the format of CHAR(<i>datetime-expression</i>) data within a QMF report. The setting controls whether QMF packages that are precompiled with the DATE(ISO) and TIME(ISO) options are used. This parameter sets the DSQAO_DSQSFISO global variable and provides an initial setting for the DSQEC_DSQSFISO global variable.</p> <p><b>Valid values</b></p> <p><b>YES</b> Results of the CHAR(<i>datetime-expression</i>) function that do not specify a format will be seen in ISO format. QMF packages that are precompiled with the DATE(ISO) and TIME(ISO) options are used.</p> <p><b>NO</b> Results of the CHAR(<i>datetime-expression</i>) function that do not specify a format will be seen in the format of the DATE FORMAT and TIME FORMAT values specified on Db2 installation panel DSNTIP4. QMF packages that are not precompiled with the DATE(ISO) and TIME(ISO) options are used. To determine the DATE FORMAT and TIME FORMAT values defined in the current Db2 subsystem, reference the DSQAO_DATE_FORMAT and DSQAO_TIME_FORMAT global variables.</p> <p><b>Default</b> YES</p> <p>The behavior set by the DSQSFISO program parameter may be changed within a QMF session by setting global variable DSQEC_DSQSFISO. The DSQSFISO program parameter setting provides the initial default value for DSQEC_DSQSFISO. Note that if DSQEC_DSQSFISO is modified, the value of DSQAO_DSQSFISO will not change. DSQEC_DSQSFISO should be referenced for the current behavior settings.</p>
DSQSIROW	F	TSO, CICS	<p><b>Controls or sets</b> Number of rows fetched from the database before the first screen of the report is displayed</p> <p><b>Valid values</b> From 0 to 99999999</p> <p><b>Default</b> 100</p> <p>For more information, see <a href="#">“Controlling report wait time”</a> on page 159.</p>

Table 32. Program parameters (continued)

Long name	Short name	Environment	Description
DSQSMODE	M	TSO, CICS	<p><b>Controls or sets</b> Mode of operation</p> <p><b>Valid values</b> B (batch operation) or I (interactive operation). Do not use this parameter when you start QMF for TSO as a Db2 for z/OS stored procedure.</p> <p><b>Default</b> I (B if QMF is started through the callable interface)</p> <p>For more information, see <a href="#">“Specifying the mode of operation (interactive or batch)”</a> on page 162.</p>
DSQSMRFI	MR	TSO, CICS	<p><b>Controls or sets</b> Use of Db2 for z/OS multirow fetch and insert features. This parameter sets the DSQAO_DSQSMRFI global variable.</p> <p><b>Valid values</b> YES (multirow fetch and insert is used) or NO (single-row fetch and insert is used). If you are using a query accelerator, verify its rowset cursor capabilities before setting this value to YES. Some query accelerators do not support queries that are run with rowset cursors, which are used by multirow fetch.</p> <p><b>Default</b> NO</p> <p>For more information, see <a href="#">“Enabling support for multirow fetch and insert”</a> on page 160.</p>
DSQSMTHD	MT	TSO	<p><b>Controls or sets</b> Use of second Db2 for z/OS thread per user. This parameter sets the DSQAO_DSQSMTHD global variable.</p> <p><b>Valid values</b> YES (use a second Db2 for z/OS thread) or NO (use a single Db2 for z/OS thread). The second thread is active only during RUN QUERY, DISPLAY TABLE commands and subsequent scrolling (BOTTOM, TOP, FORWARD, BACKWARD, RIGHT, and LEFT) of reports with open cursors. Commands that make use of a complete report such as EXPORT DATA, SAVE DATA, PRINT REPORT, and DPRES will also use the second thread.</p> <p><b>Default</b> NO</p>

Table 32. Program parameters (continued)

Long name	Short name	Environment	Description
DSQSPILL	L	TSO, CICS	<p><b>Controls or sets</b> Use of extra storage to hold data that is no longer needed in active storage</p> <p><b>Valid values</b> YES (uses spill storage for data no longer needed in active storage) or NO (does not use spill storage)</p> <p><b>Default</b> YES when QMF is running under TSO; NO when QMF is running under CICS. When DSQSPILL is YES, use DSQSPTYP to specify the type of storage to be used for spill data.</p> <p>For more information, see <a href="#">“Acquiring extra storage for data no longer needed in virtual storage”</a> on page 152.</p>
DSQSPPLAN	P	TSO	<p><b>Controls or sets</b> Name of QMF application plan</p> <p><b>Valid values</b> Any valid Db2 application plan name. This parameter is required in the DSQSCMDn exec when you start QMF for TSO as a Db2 for z/OS stored procedure.</p> <p><b>Default</b> QMF12</p> <p>For more information, see <a href="#">“Specifying the name of the QMF application plan”</a> on page 149.</p>
DSQSPRID	U	TSO	<p><b>Controls or sets</b> QMF profile key for QMF for TSO installations</p> <p><b>Valid values</b> PRIMEID (QMF authenticates the user's Db2 primary authorization ID against the CREATOR column of the Q.PROFILES table) or TSOID (QMF authenticates the user's TSO logon ID against the CREATOR column of the Q.PROFILES table). Only a value of PRIMEID is allowed when you start QMF for TSO as Db2 for z/OS stored procedure.</p> <p><b>Default</b> PRIMEID</p> <p>For more information, see <a href="#">“Specifying which ID to use as the QMF profile key under TSO”</a> on page 150.</p>



Table 32. Program parameters (continued)

Long name	Short name	Environment	Description
DSQSPTYP	ST	TSO	<p><b>Controls or sets</b> Use of extended storage to hold data that is not needed in active storage; supported only when the DSQSPILL parameter is set to YES</p> <p><b>Valid values</b> FILE (spills data to a file) or 64BIT (spills data to extended virtual storage)</p> <p><b>Default</b> FILE</p> <p>For more information, see <a href="#">“Spilling report data to extended virtual storage (TSO only)”</a> on page 152.</p>
DSQSRSTG	R	TSO	<p><b>Controls or sets</b> Number of bytes of storage that you want to reserve from the TSO region to be used for applications other than QMF</p> <p>When QMF generates the first report in a QMF session, it subtracts the amount of storage specified by this parameter from the amount of free continuous storage available to QMF at the time. The remaining storage is then available for QMF report operations for the remainder of the QMF session.</p> <p><b>Valid values</b> From 0 to 999999999</p> <p><b>Default</b> 0</p> <p>For more information, see <a href="#">“Reserving virtual storage from the TSO region to be used for applications other than QMF”</a> on page 151.</p>
DSQSRUN	I	TSO, CICS	<p><b>Controls or sets</b> Name of QMF procedure to run when QMF starts, before the QMF home panel is displayed</p> <p><b>Valid values</b> Any valid procedure name. Do not use this parameter when you start QMF for TSO as a Db2 for z/OS stored procedure.</p> <p><b>Default</b> No initial procedure is run</p> <p>For more information, see <a href="#">“Specifying an initial procedure to run when QMF starts”</a> on page 163.</p>

Table 32. Program parameters (continued)

Long name	Short name	Environment	Description
DSQSSPQN	—	CICS	<p><b>Controls or sets</b> Name of CICS queue to hold data no longer needed in active storage when DSQSPILL has been set to YES</p> <p><b>Valid values</b> Any name that follows CICS queue-naming conventions</p> <p><b>Default</b> DSQSnnnn (where nnnn is the CICS terminal ID)</p> <p>For more information, see <a href="#">“Spilling report data to a file in CICS”</a> on page 157.</p>
DSQSSUBS	S	TSO	<p><b>Controls or sets</b> Name of Db2 subsystem in which to start QMF under TSO</p> <p><b>Valid values</b> Any valid Db2 subsystem name. This parameter is required in the DSQSCMDn exec when you start QMF for TSO as a Db2 for z/OS stored procedure.</p> <p><b>Default</b> DSN</p> <p>For more information, see <a href="#">“Specifying the name of the Db2 for z/OS subsystem in which to start QMF under TSO”</a> on page 148.</p>

## Setting database and environment parameters

These program parameters control settings and preferences related to the database and environment in which QMF runs.

### Specifying the name of the Db2 for z/OS subsystem in which to start QMF under TSO

QMF must be started in a Db2 for z/OS subsystem running under TSO. Use the DSQSSUBS parameter to specify the name of the Db2 for z/OS subsystem in which to start QMF.

In a data-sharing environment, you can use the Db2 group attachment name in place of a Db2 subsystem name.

A value for this parameter is required in the DSQSCMDn exec when you start QMF for TSO as a Db2 for z/OS stored procedure.

#### Related concepts

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

#### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## **Specifying the initial database connection**

You can start QMF in the local Db2 for z/OS subsystem and connect immediately to another database before the QMF home panel is displayed.

### **Procedure**

To start QMF and specify the initial database connections, follow these steps:

1. Set the DSQSSUBS parameter to the name of the Db2 for z/OS subsystem in which QMF is started.
2. Set the DSQSDBNM parameter to the name of the location to which you want to connect before the QMF home panel is displayed. Location names can be up to 16 characters.

### **Results**

When QMF for TSO is started as a Db2 for z/OS stored procedure, this parameter is ignored if it is specified in the DSQSCMDn exec that passes parameter values to the stored procedure interface.

### **Related concepts**

[Specifying the name of the Db2 for z/OS subsystem in which to start QMF under TSO](#)

QMF must be started in a Db2 for z/OS subsystem running under TSO. Use the DSQSSUBS parameter to specify the name of the Db2 for z/OS subsystem in which to start QMF.

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### **Related tasks**

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## **Specifying the name of the QMF application plan**

The default name for the QMF application plan is QMF12. If you have chosen a different name for the QMF application plan, you must pass the correct name on this parameter.

A value for this parameter is required in the DSQSCMDn exec when you start QMF for TSO as a Db2 for z/OS stored procedure.

### **Related concepts**

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### **Related tasks**

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

#### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

### **Specifying which ID to use as the QMF profile key under TSO**

If you choose to create specific profiles for each user in QMF for TSO, the values in the CREATOR column of the table can all be TSO logon IDs or Db2 primary authorization IDs. If you choose to have QMF authenticate users using their TSO logon IDs, you must specify that choice when you start QMF.

When a user starts QMF, QMF authenticates the user ID under which the program was started with the value in the CREATOR column of the Q.PROFILES table. QMF first checks the CREATOR column for the specific user ID under which the program was started. If no row in the table corresponds to that ID, QMF checks for a row in the Q.PROFILES table where CREATOR=SYSTEM.

If you set QMF to authenticate users using their TSO logon IDs, enter a value of TS0ID for the DSQSPRID program parameter when you start QMF:

```
DSQSPRID=TS0ID
```

A value of PRIMEID is the default for this parameter, indicating that QMF will authenticate users using their Db2 primary authorization IDs.

Only the value PRIMEID is valid in the DSQSCMD $n$  exec when you start TSO as a Db2 for z/OS stored procedure.

#### **Related concepts**

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

#### **Related tasks**

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

#### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## **Defining storage for reports**

---

When a user performs a QMF task that retrieves, imports, or exports data, the data is held in storage until the operation is complete. In both CICS and TSO, you can set a fixed upper limit for the amount of virtual storage, also called active storage, to be used for these operations. In TSO, you can also specify a percentage or portion of available storage to be used for reports.

#### **About this task**

To improve QMF performance in both TSO and CICS, you can also specify additional storage to be used to hold data no longer needed in active storage; this type of storage is called spill storage. You can spill data to extended virtual storage or a file.

## Defining a fixed amount of virtual storage for reports

Use the DSQSBSTG parameter to specify a fixed number of bytes of virtual storage to be used by QMF for report operations.

Set the DSQSBSTG parameter to a value between 0 and 2147483647. This value represents the maximum number of bytes of virtual storage to be used for report operations in QMF. If you set this parameter to a value less than 15000, 15000 is used automatically.

Defaults for the DSQSBSTG parameter are as follows:

- 0 when you start QMF from TSO, ISPF, or native z/OS
- 500000 when you start QMF from CICS

### Related reference

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## Defining a variable amount of virtual storage for reports

In TSO, you can define the amount of virtual storage that is available for reports as a percentage or portion of the total available storage.

With the DSQSBSTG program parameter, you can directly define the storage that is used for reports as a percentage of the available storage. With the DSQSRSTG program parameter, you can define the amount of storage that is reserved for applications other than QMF. The remaining amount of virtual storage is available for report operations.

### Defining storage for reports as a percentage of available storage

In TSO only, you can define the amount of storage that is available for reports as a percentage of the total available virtual storage.

#### Procedure

Set the DSQSBSTG parameter to a value in the range 1% - 100%.

A percent sign (%) must follow the numeric value with no space between the value and the sign.

### Reserving virtual storage from the TSO region to be used for applications other than QMF

Use the DSQSRSTG parameter in QMF for TSO to specify the maximum amount of virtual storage that you want to reserve in the TSO region for use by applications other than QMF, such as TSO commands, REXX, or ISPF. This reserved storage is then not used for QMF report operations.

#### About this task

The amount of storage that you set on this parameter can affect other programs and the generation of reports. When QMF creates a new report, it determines the amount of storage that is available in the QMF address space. QMF subtracts the amount of storage that you specify for the DSQSRSTG parameter from the amount of free storage available to QMF at the time. The remaining amount of virtual storage is available for report operations.

#### Procedure

To set the amount of storage to reserve from the TSO region for applications other than QMF, follow these steps:

1. Ensure that the DSQSBSTG parameter is set to 0, which is the default value.

If you set the DSQSBSTG to a non-zero value in QMF for TSO, QMF allocates the specified amount of storage and ignores the DSQSRSTG parameter.

2. Set the DSQSRSTG parameter to a value between 0 and 99999999.

This value represents the maximum number of bytes of virtual storage that are reserved for applications not related to QMF.

If you specify 0 for both the DSQSBSTG and DSQSRSTG parameters, no storage is reserved for applications other than QMF. This value is probably adequate for users who never use z/OS, TSO commands, REXX, ISPF or other services not related to QMF during QMF sessions. However, be aware that even the most casual users might unknowingly use a program not related to QMF when they issue site-defined QMF commands. Such commands are performed by QMF applications, which generally make extensive use of programs not related to QMF. Take this into account when selecting values for DSQSRSTG and DSQSBSTG.

### **Related concepts**

[Defining a fixed amount of virtual storage for reports](#)

Use the DSQSBSTG parameter to specify a fixed number of bytes of virtual storage to be used by QMF for report operations.

### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## **Acquiring extra storage for data no longer needed in virtual storage**

Because demands on active storage can affect the operation of other programs, QMF allows you to specify extra storage (called spill storage) to store data no longer needed in active storage.

### **About this task**

When a QMF report is returned or a user issues an IMPORT or EXPORT command, the data required by the operation is held in virtual storage, also called active storage. Using spill storage can improve performance in an interactive QMF session because QMF does not need to return to the database for multiple copies of the same data. Instead, data is retrieved from spill storage and is brought back into active storage as needed.

Data is written to spill storage as needed until one of the following events occurs:

- The QMF report is reset (through the RESET DATA command) or a new report is created (for example, when a user issues a QMF RUN QUERY, DISPLAY TABLE, or IMPORT DATA command).
- Your query has finished (all requested rows have been retrieved) and the DATA object is therefore complete.
- The maximum amount of storage that you specified for spill data has been used.

There are two types of spill storage. In QMF for TSO, you can use extended storage for spill data. In both QMF for TSO and QMF for CICS, you can allocate a file for spill data to be used when active storage is exhausted.

### **Related tasks**

[Spilling report data to extended virtual storage \(TSO only\)](#)

In QMF for TSO, use extended storage for spill data unless the system on which QMF is running has very limited extended storage available.

### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

### **Spilling report data to extended virtual storage (TSO only)**

In QMF for TSO, use extended storage for spill data unless the system on which QMF is running has very limited extended storage available.

### **About this task**

## Procedure

To spill report data to extended virtual storage in QMF for TSO, follow these steps:

1. Set the following program parameters when you start QMF:
  - DSQSPILL=YES
  - DSQSPTYP=64BIT
2. Set the DSQEC\_EXTND\_STG global variable to the number of megabytes of extended storage for QMF to acquire on each request to the extended storage manager.

When a user performs an operation that requires extended storage, QMF issues repeated requests to the extended storage manager for the specified amount until the operation is complete or extended storage is exhausted. The default value for this global variable is 25, indicating that QMF requests 25 MB of storage on each request.

When setting this global variable, consider the average size of DATA objects with which your QMF users work. If the average size is very large and you set the DSQEC\_EXTND\_STG variable too low, QMF must issue many calls to the extended storage manager to complete the DATA object, which could affect overall performance.

## Spilling report data to a file in TSO

If QMF users are working with data types that require larger amounts of virtual storage, such as XML or LOB data types, you must configure QMF to spill data to extended storage rather than a file.

## About this task

When the QMF report is reset (through the RESET DATA command) or a new report is created (for example, when a user issues a RUN QUERY, DISPLAY TABLE, or IMPORT DATA command), QMF resets the space allocation to the primary allocation by opening and closing the spill file with a TTR of zero.

## Related concepts

### [Troubleshooting spill file problems](#)

Creating a spill file can solve storage problems, but it can also cause other problems. You can usually recover from these problems by adding space, moving the user's spill file, and making changes that reviewing factors that affect performance.

## Estimating the space required for a spill file

Providing enough spill storage is important because, when spill storage is exhausted, QMF does not use the data already retrieved into the spill file, but instead retrieves it again from the database, using virtual storage to hold it.

## About this task

## Procedure

Use the following procedure to calculate the amount of space required for the spill file:

1. Calculate the width (W) of the longest row that can appear in the DATA object by adding field widths in bytes (use the following table).

Defined columns and any columns containing LOB or XML data do not get written to the spill file. LOB and XML columns are stored in extended storage when they are fetched and moved to report storage when they are used.

Field type	Field width in bytes
CHAR( <i>n</i> ), BINARY( <i>n</i> )	<i>n</i> +2
VARCHAR( <i>n</i> ), VARBINARY( <i>n</i> )	<i>n</i> +4

*Table 33. Widths of fields for different data types. Use this table to help estimate the width of a row of data when estimating the required size of the spill file. (continued)*

Field type	Field width in bytes
DATE	12
DECFLOAT(16)	8
DECFLOAT(34)	16
DECIMAL( <i>n,m</i> )	$(n+1)/2+2$ , <i>n</i> odd $(n+2)/2+2$ , <i>n</i> even
FLOAT(21)	10
FLOAT(53)	10
GRAPHIC( <i>n</i> )	$n*2+2$
INTEGER	6
SMALLINT	4
BIGINT	22
TIME	10
TIMESTAMP(0)	21
TIMESTAMP( <i>n</i> )	$22 + n$ (where <i>n</i> = 1 to 12)
TIMESTAMP(0) WITH TIME ZONE	29
TIMESTAMP( <i>n</i> ) WITH TIME ZONE	$30 + n$ (where <i>n</i> = 1 to 12)
LONG VARCHAR	(depends on other field lengths)
LONG VARGRAPHIC	(depends on other field lengths)
VARGRAPHIC( <i>n</i> )	$n*2+4$

If a row contains LONG VARCHAR or LONG VARGRAPHIC fields, space is first allotted for all other fields. Then the remaining space is divided by the number of fields, and each LONG VARCHAR or LONG VARGRAPHIC field is truncated to that length.

2. Use the value of *W* to calculate either the number of rows per page (*R*) or the number of pages per row (*P*):
  - If *W* is 4096 or less, calculate the number of rows per page (*R*) using  $R = 4096/W$ , and round the result down to the next lowest integer. When *W* is 4096 or less, QMF fits as many rows as it can into a page, without spanning pages.
  - If *W* is greater than 4096, calculate the number of pages per row (*P*), using  $P = W/4096$ , and round up to the next highest integer. When *W* is greater than 4096, QMF uses the minimum number of pages to hold a row, spanning pages regardless of column boundaries. Each row begins at the start of a page.
3. Use values from step “1” on page 153 and step “2” on page 154 to calculate the number of pages required for the spill file:
  - If *W* is 4096 or less, calculate the number of pages required for the spill file by dividing the number of rows in the table by *R*.
  - If *W* is greater than 4096, calculate the number of pages required for the spill file by multiplying the number of rows in the table by *P*.

The following table shows a sample calculation for an individual spill file.



Table 34. Sample row width calculation for a spill file		
Content of row	Calculation	Contribution to width
Two SMALLINT columns	$2 \times 4 =$	8 bytes
One INTEGER column		6 bytes
One DECIMAL(3,2) column	$(3+1)/2+2 =$	4 bytes
One DECIMAL(6,0) column	$(6+2)/2+2 =$	6 bytes
One FLOAT column		10 bytes
One CHAR(10) column	$10 + 2 =$	12 bytes
One VARCHAR(16) column	$16 + 4 =$	20 bytes
<b>Total width of row</b>		<b>59 bytes</b>

The following sample calculations provide two ways to calculate the spill file space.

When R=4096/540, 7 rows/buffer:

$$\frac{600,000 \text{ rows}}{7} * \frac{1 \text{ track}}{10 \text{ blocks}} * \frac{1 \text{ cylinder}}{15 \text{ tracks}} = 571 \text{ cylinders}$$

When R=6000, 2 buffers/row:

$$6000 \text{ rows} * 2 \text{ blocks/row} * \frac{1 \text{ track}}{10 \text{ blocks}} * \frac{1 \text{ cylinder}}{15 \text{ tracks}} = 800 \text{ cylinders}$$

4. Estimate the number of concurrent QMF users and provide enough spill storage to accommodate the individual spill files of all of these users as well as any other storage demands from other programs.

### Related tasks

Spilling report data to extended virtual storage (TSO only)

In QMF for TSO, use extended storage for spill data unless the system on which QMF is running has very limited extended storage available.

### Allocating a spill file

You can allocate a spill file through a DD statement in the user's logon procedure, JCL, or CLIST.

### Before you begin

Ensure that the following program parameter values are set when you start QMF:

- DSQSPILL=YES
- DSQSPTYP=FILE

Both of these values are the default values for these program parameters.

### About this task

**Important:** QMF performance might slow down if QMF needs a data row (as a result of a SCROLL BACKWARD command) and that data is not in the spill file or in virtual storage. In this case, an I/O abend occurs. QMF provides error-handling routines for the DCB SYNAD exit and recovers from these I/O abends by using information that is provided by the DCB abend exit. QMF then stops using the spill file and fetches the data again from the database. If you use z/OS tools that intercept DCB abends (such as the B37 abend), ensure that you exclude the QMF spill file from such operations. Otherwise, QMF cannot properly manage the spill file, causing not only unpredictable results, but also difficulty in tracing and diagnosing any errors.

## Procedure

When you allocate the spill file, follow these guidelines:

- Allocate the file as a temporary data set, which persists during the user's session.
- If the DASD is a single volume that does not exceed 16,777,215 tracks, you can allocate the file to a virtual I/O device. This allocation is shown on the UNIT=SYSVIO statement in the example allocation statement. Or you can allocate the spill file to other DASD storage. You cannot allocate the spill file to a terminal or display device.

The QMF spill file is allocated for direct access by using the BSAM access method. The z/OS NOTE, POINT, READ, WRITE, and CHECK services are used to manage the QMF spill file. Any z/OS restrictions that apply to the use of these system services also apply to the use of the QMF spill file.

- Specify fixed-length records, one record for each block. The records must always be unblocked. (A block is the size of a z/OS page: 4096 bytes.)
- Specify release storage by using the RLSE keyword as shown in the following example.
- To allocate a spill file in a CLIST, issue an ATTR statement followed by an ALLOC statement as in the following example.

The ATTR statement is not required, but can be specified.

```
ATTR SPILL RECFM(F) LRECL(4096) BLKSIZE(4096)
ALLOC FILE(DSQSPILL) UNIT(SYSVIO) SPACE(10,20) RELEASE +
NEW DELETE USING(SPILL)
```

- If you must allocate or reallocate the spill file during a user's QMF session, follow these steps:
  - a) Issue the QMF RESET DATA command to close the current spill file.
  - b) Issue the FREE command to free the current spill file allocation.

If you issue this command from within QMF, it must be preceded by the QMF TSO command, as in the following example:

```
-----1=Help      2=List      3=End      4=Show      5=Chart      6=Query
7=Retrieve   8=Edit Table 9=Form    10=Proc    11=Profile   12=Report
OK, you may enter a command.
COMMAND ==> TSO FREE FILE(DSQSPILL)
```

- c) Allocate the new spill file by using ATTR and ALLOC statements as in the earlier example. When the user creates a report, QMF uses the new spill file specifications.

## Example

An example allocation statement for spill storage appears in the sample logon procedure that is provided with QMF, where the spill file is allocated to the ddname DSQSPILL.

```
//DSQSPILL DD DSN=&&SPILL,DISP=(NEW,DELETE),
//          UNIT=SYSVIO,SPACE=(TRK,(10,20),RLSE),
//          DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
```

The SPACE operand in this statement can minimize spill file storage requirements during a session:

- The small primary extent keeps the space that is held by the spill file to 10 tracks during sessions when a spill file is not needed.
- The much larger secondary extents are used only when a spill file is required.
- The RLSE keyword lets QMF release all secondary extents when the user's DATA object is reset. For example, the DATA object can be reset when the user runs a new query and a new report is returned.

## Spilling report data to a file in CICS

In CICS, spill data must go to a temporary storage queue.

### Procedure

To specify the use of a file for spilling report data in CICS, follow these steps:

1. Estimate the space required for the spill file. See [“Estimating the space required for a spill file”](#) on page 153.
2. Enlarge DFHTEMP storage to accommodate the individual spill files of all concurrent QMF users as well as demands from transactions other than QMF.
3. Set the following program parameters when you start QMF:
  - DSQSPILL=YES
  - DSQSPTYP=FILE
  - DSQSSPQN=*queue name*

### Results

For example, to specify the name MYDATA for the CICS queue when you start QMF in English, enter:

```
QMFE DSQSPILL=YES, DSQSPTYP=FILE, DSQSSPQN=MYDATA
```

### Spilling report data when QMF is running in batch mode

Spill storage is most useful for improving performance in an interactive QMF session, when the DSQSMODE parameter is set to I. However, if you are running QMF noninteractively (the DSQSMODE parameter is set to B), using spill storage can also improve performance when multiple passes of the data are required to produce the report.

Spill storage might also be necessary to complete the DATA object (for example, when a RUN QUERY command is followed by a SAVE DATA command).

Multiple passes of the data are required in these conditions:

- You must print several reports with different formats for the same data.
- You use PCT, CPCT, TCPCT, or TPCT edit codes with the report.
- You print a report that requires QMF to split the pages, because the report is wider than the print width.

When QMF is running in batch mode, the DSQSPILL program parameter should be set based on the work to be done. If the batch job will produce a large DATA object for printing, then allocating a spill file can have a negative effect on performance. In these cases, specify DSQSPILL=NO when starting QMF.

If you start a batch QMF session from within a CICS application and you choose not to specify a CICS terminal ID, you must explicitly specify a value for the DSQSSPQN parameter or QMF does not start.

### Troubleshooting spill file problems

Creating a spill file can solve storage problems, but it can also cause other problems. You can usually recover from these problems by adding space, moving the user's spill file, and making changes that reviewing factors that affect performance.

For more information, see one of the following sections:

- [“Problems caused by too little space on a DASD volume”](#) on page 157
- [“Problems caused by increasing the secondary space allocation for the spill file”](#) on page 158
- [“Performance considerations”](#) on page 158

### Problems caused by too little space on a DASD volume

If several users with the same QMF logon procedure are experiencing spill file problems, and their common logon procedure allocates all their spill files to a single specific DASD volume, the problems

could be due to insufficient space on this volume. If this is the case, you can solve the problem by changing the spill file DD statement in their logon procedure. The new DD statement might make a nonspecific volume reference instead of referring to a specific volume.

### **Problems caused by increasing the secondary space allocation for the spill file**

Increasing the spill file's secondary allocation could solve one user's spill file problems, but in doing so, you might create spill file problems for others. If you must increase the secondary allocation, consider moving the user's spill file to a volume not used for other users' spill files.

A user can unknowingly create spill file problems for others. For example, a user might scroll to the bottom of a large report and overflow the spill file, but do nothing to bring about the incomplete data condition. This would be true if the user failed to issue certain types of commands between the time the report was first displayed and the time it was replaced by another report. In the interim, the user's spill file might unnecessarily hold space that others need.

### **Performance considerations**

If you are not using conditional formatting or column definitions (which use REXX and have additional performance considerations), the performance you observe is the result of accessing data in the database.

Part of the processing time is devoted to writing the data to the spill file so that it can be fetched later. If you have enough storage available to QMF after your data is retrieved the first time, QMF will not need to reaccess the database to obtain rows a second time.

Performance is affected by several factors:

- The value of the DSQSIROW parameter (initial number of rows to fetch before displaying the first screen of the report).
- Whether you perform a task that requires multiple passes of the data. (Certain usage codes, such as PCT, require that all the data be read before the first report screen displayed.) This primarily affects the initial display of the report only.
- The amount of memory required to hold one row of data.
- Where the data is retrieved from when additional passes are required to complete the report:
  - The database (not all data fits in memory and the spill file)
  - Virtual memory and the spill file
  - Virtual memory alone
- Whether you are scrolling backward or forward. Successive FORWARD commands usually perform best. BACKWARD commands might require starting over at the start of the DATA object. This depends on the amount of memory, how far back you want to scroll, and the complexity of the report.

For very large reports with little memory and insufficient DSQSPILL allocation, the entire DATA object could be read from row 1 to the new current row, every time the BACKWARD command is used.

The best performance is achieved when there is sufficient memory to hold all data and DSQSPILL is not used. If the value for the DSQSIROW parameter is high enough to allow the entire DATA object to be retrieved into virtual memory before the first screen of the report is displayed, the database locks are released, providing better performance in scrolling the displayed report. Releasing the locks could also improve performance for other users. However, setting DSQSIROW higher can also slow the display of the first screen of the report.

To improve performance with commands following RUN QUERY commands that produce large answer sets, or to avoid storage issues when using the SAVE DATA command for large answer sets, you can use multiple database threads. To run end user SQL commands on a separate database thread, this optional second thread is controlled by the QMF program parameter DSQSMTHD (YES/NO; default: NO). This program parameter allows QMF Administrators to control whether QMF uses an additional database thread or executes on a single thread as it does by default. Note: The use of 2 threads per user can affect general Db2 resource consumption (CTHREAD limits, etc).

## Related concepts

### Controlling report wait time

The DSQSIROW parameter controls the number of rows the database fetches before QMF displays the first report screen.

## Controlling performance of fetch and insert operations

---

Two program parameters in QMF help you control database performance during fetch and insert operations.

### Controlling report wait time

The DSQSIROW parameter controls the number of rows the database fetches before QMF displays the first report screen.

Use the DSQSIROW parameter to specify the maximum number of rows that QMF retrieves into the DATA object before displaying the first screen of the report to the user. DSQSIROW applies only to the initial load of a new DATA object, created by:

- Running queries that use SQL SELECT statements
- Displaying a database table with the QMF DISPLAY command

The value of DSQSIROW determines when QMF stops fetching rows from the database and displays the report. However, QMF always retrieves enough rows to fill a 4 KB buffer prior to displaying the first screen of data, regardless of the DSQSIROW value. For example, suppose that 62 rows are required to fill a buffer and you set DSQSIROW to 50. QMF retrieves 62 rows of data and, upon comparing 62 to 50, stops retrieving rows and displays the first screen of data.

Some report formatting options, such as PCT and ACROSS usage codes, require that all the data be retrieved before QMF displays the first screen. QMF ignores the DSQSIROW value in these situations.

### Performance with small DSQSIROW values

Do not use DSQSIROW to limit the number of rows that QMF displays on the screen. Although you can specify a small value, QMF retrieves enough rows to fill the screen display in an interactive session. If you use too small a value for the DSQSIROW parameter, QMF might not be able to complete the DATA object before the first screen of the report is displayed. An incomplete DATA object causes share locks on the data, which can prevent other users from updating the data. If necessary, you can avoid this problem by allowing uncommitted read.

Many users might be affected if a QMF control table or a part of the system catalog is locked. You can release the locks in one of the following ways:

- Use the BOTTOM command to retrieve the remaining rows into the DATA object, then release the locks.
- Use the RESET DATA command to release these locks and clear the DATA object, whether or not all requested rows have been retrieved.
- Use any SAVE command (for example, SAVE DATA or SAVE FORM) to retrieve and save the remaining rows into the DATA object, then release the locks.

To get the best performance in a batch session (when the DSQSMODE parameter is set to B), use a value of 0 for DSQSIROW unless you want to minimize the number of open read locks while QMF is retrieving or formatting data.

### Performance with large DSQSIROW values

If you use too large a value for the DSQSIROW parameter, QMF might take a long time to display the first screen of data. Additionally, if there is not enough storage to retrieve the number of rows specified by DSQSIROW, storage becomes full. If storage is full and you are running QMF in CICS, CICS waits for storage to become available. In TSO, QMF issues a message indicating that storage is full.

## Related concepts

[Allowing uncommitted read](#)

If you want your QMF session to allow uncommitted read, you can specify a value for the global variable DSQEC\_ISOLATION in the Q.SYSTEM\_INI procedure.

## Related reference

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## Enabling support for multirow fetch and insert

The DSQSMRFI parameter controls whether the database uses multirow or single-row fetch and insert.

Use the DSQSMRFI parameter (MR) to control whether QMF uses multirow processing for database fetch and insert operations. With multirow processing, a single SQL statement can process several rows of data, resulting in decreased network traffic. The following commands take advantage of increased performance from multirow support:

- BOTTOM
- FORWARD
- DISPLAY TABLE
- DPRE
- EXPORT DATA and EXPORT TABLE
- IMPORT TABLE
- PRINT REPORT and PRINT TABLE
- RUN QUERY or RUN PROC (when these specify retrieval operations)
- SAVE DATA

Single-row fetch is automatically used in the following situations:

- When the command requires processing of XML or LOB data
- When you start QMF in the requester database with MR=YES, use the QMF CONNECT command to connect to a server other than Db2 for z/OS Version 8.1.5 (or later), then proceed to issue a command that requires fetch or insert operations

### Restriction:

- If MR is set to YES and you use a QMF command that includes a three-part name, both the requester where the command is initiated and the server to which the command is directed must be Db2 for z/OS 8.1.5 (or later) or the command fails. Start QMF with MR=NO if you will be issuing QMF commands with three-part names directed to servers other than Db2 for z/OS Version 8.1.5 (or later). Commands with three-part names cannot be directed to DB2 for VSE and VM servers.
- Depending on the capabilities of your query accelerator, the QMF DSQSMRFI (MR) program parameter might be of importance. Some query accelerators do not support queries that are run with rowset cursors. If QMF is started with the DSQSMRFI (MR) program parameter set to YES, QMF uses a rowset cursor. If your query accelerator does not support queries that are run with rowset cursors, start QMF with MR=NO.
- QMF supports operations with XML data only when you are connected to a database release that supports the XML data type.

## Related reference

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## Using multiple database threads

---

To improve performance with commands following RUN QUERY commands that produce large answer sets, or to avoid storage issues when using the SAVE DATA command for large answer sets, you can run end user SQL commands on a separate database thread.

### About this task

To use multiple threads, the optional second thread is controlled by the QMF program parameter DSQSMTHD (YES/NO; default: NO). This program parameter allows QMF Administrators to control whether QMF uses an additional database thread or executes on a single thread as it does by default.

The end user SQL that uses the second thread includes the following QMF commands:

- RUN QUERY (except for RUN QUERY with the TABLE keyword)
- DISPLAY TABLE
- Commands that scroll reports (BOTTOM, TOP, FORWARD, BACKWARD, RIGHT and LEFT)
- Commands that require complete reports (EXPORT DATA, SAVE DATA, PRINT REPORT, DPRE, CONNECT, ERASE TABLE)

When DSQSMTHD is set to YES, the value of the DSQAO\_CURSOR\_OPEN state global variable will reflect the status of the end user SQL second thread.

DSQSMTHD is applicable to QMF running under TSO (QMF CAF interface) only. DSQSMTHD settings are ignored when running QMF under CICS and running QMF under the stored procedure interface.

When DSQSMTHD is set to YES, the effect of global variable DSQEC\_RESET\_RPT (prompting if an incomplete report is outstanding) is changed. When you CONNECT to remote databases, or CONNECT to a new userid and password within the same data base, the second thread will also connect. Any open report objects must be completed prior to the CONNECT. DSQEC\_RESET\_RPT can be used to alert you to this situation.

While the ERASE TABLE command does not utilize the user thread, it continues to require that an incomplete data object be completed; this prevents deadlock and timeout situations.

In the case of an incomplete report (with DSQSMTHD=YES), the list of commands that will COMMIT the report (release locks) and delete the thread is:

- RESET DATA
- RUN QUERY (SELECT statements)
- PRINT TABLE
- DISPLAY TABLE
- EXPORT TABLE
- CONNECT
- ERASE TABLE

Incomplete data prompting (DSQEC\_RESET\_RPT) will also occur (except for RESET DATA).

**Note:** The use of two threads per user can affect general Db2 resource consumption (CTHREAD limits, etc).

### Related concepts

[Troubleshooting spill file problems](#)

Creating a spill file can solve storage problems, but it can also cause other problems. You can usually recover from these problems by adding space, moving the user's spill file, and making changes that reviewing factors that affect performance.

### **Related tasks**

[Using multiple database threads](#)

To improve performance with commands following RUN QUERY commands that produce large answer sets, or to avoid storage issues when using the SAVE DATA command for large answer sets, you can run end user SQL commands on a separate database thread.

## **Automating QMF activity**

---

The DSQSMODE and DSQSRUN parameters can automate QMF activity and save resources and time.

### **Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## **Specifying the mode of operation (interactive or batch)**

Using the program parameter DSQSMODE, you can save resources and time by starting a batch session to perform your work in QMF. Your system is then free for you to do other work while the transaction is running.

### **Before you begin**

Do not specify this parameter when you start QMF for TSO as a Db2 for z/OS stored procedure; a value of B, indicating batch mode, is already specified for you in the DSQSCMDn exec that starts the interface.

### **About this task**

Some query and report-writing tasks that users must perform might not require interaction with QMF. For example, a salesperson might use the same QMF procedure every few days to query a set of tables for account status. Although the data changes, the procedure and tasks required to access the data remain the same.

### **Procedure**

To start QMF in batch mode, follow these steps:

1. Specify a value of B on the DSQSMODE parameter.

To specify interactive operation of QMF, use a value of I on the DSQSMODE parameter. This is the default value.

2. Because a batch session displays no QMF panels, use the DSQSRUN parameter to run an initial procedure that does the required work and exits the program

For example, the following command starts a batch session in English that runs an initial procedure called STARTPROC, which is owned by user JONES.

```
DSQQMFE M=B,I=JONES.STARTPROC
```

3. If necessary, add the DSQSDBNM parameter to specify the initial database to which you want to connect if you do not want to use the default database location.

### **Related concepts**

[Specifying an initial procedure to run when QMF starts](#)

The DSQSRUN parameter passes the name of a QMF procedure that runs as soon as QMF starts and then exits QMF.

[Starting QMF as a Db2 for z/OS stored procedure](#)



The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## Specifying an initial procedure to run when QMF starts

The DSQSRUN parameter passes the name of a QMF procedure that runs as soon as QMF starts and then exits QMF.

For example, to start QMF in English and run an initial procedure named STARTPROC, issue the following command:

```
DSQQMFE I=STARTPROC
```

Qualify the procedure name with the SQL authorization ID of its owner if other users are using it to start QMF. For example, if user JONES owns the STARTPROC procedure, enter:

```
DSQQMFE I=JONES.STARTPROC
```

When you pass the name of an initial procedure, QMF issues a RUN PROC command, which runs the procedure that you specify.

**Important:** QMF does not allow blanks in the user ID and procedure name syntax. For example, QMF doesn't recognize the following procedure name:

```
DSQQMFE I=JONES. STARTPROC
```

To use a procedure name with an embedded blank, you must enclose the name in quotes:

```
DSQQMFE I=JONES. 'START PROC'
```

- You can use the DSQSRUN parameter to:
- Automate QMF activity in batch mode so that you can conserve resources that are normally used when running interactively.
- Allow users to perform interactive QMF work within the confines of a predefined procedure, then exit when they are finished with the work specified in the procedure.

Do not specify this parameter when you start QMF for TSO as a Db2 for z/OS stored procedure; when you start QMF in this manner, you use the CALL statement to pass the name of an initial query or procedure name to be run when QMF starts.

### Related concepts

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### Running an initial procedure in batch mode

You can run a procedure in batch mode by using a value of B for the DSQSMODE parameter and naming a procedure using the DSQSRUN parameter.

For example, suppose that every Monday morning, you must produce an inventory status report. Each Sunday night, you must run a query that retrieves data from the same columns of a table called INVENTORY. Your query might look something like the following query.

```
SELECT * FROM INVENTORY
WHERE STOCK < 20
```

For this example, we will name this query INVENTORY\_QUERY. The procedure that you use to run this query and print the status report might look something like the following procedure. For these examples, we'll call this QMF procedure INVENTORY\_PROC:

```
RUN QUERY INVENTORY_QUERY
PRINT REPORT
EXIT
```

The procedure includes an EXIT command because, when QMF is running in batch mode, no user is present to end the QMF session. EXIT ends the QMF session and frees the resources being held by QMF. Always use an EXIT command in an initial procedure that runs in batch mode.

Because the tasks involved in creating the report do not change (only the data changes), you can use the DSQSRUN parameter to query the INVENTORY table off-shift on Sunday night and print the report:

```
QMFn I=INVENTORY_PROC,M=B
```

### Performing interactive QMF work with an initial procedure

You can use an initial procedure in an interactive QMF session to predefine QMF tasks for end users, making it easy for them to access only the data they need.

For example, suppose that a QMF end user has the responsibility of producing an inventory status report every Monday morning. The user might know the value that indicates low stock but might not know exactly how to produce the status report. In this case, you can put a variable in the query so that the user needs to enter only the value that indicates low stock. We will call this query INVENTORY\_QUERY:

```
SELECT * FROM INVENTORY
WHERE STOCK < &LOWSTOCK
```

Because the user might want to view the data before printing it, your INVENTORY\_PROC procedure might not include the EXIT command:

```
RUN QUERY INVENTORY_QUERY
```

You can then use the DSQSRUN parameter without specifying the DSQSMODE parameter, so that you start an interactive session for the user:

```
QMFn I=INVENTORY_PROC
```

The INVENTORY\_PROC procedure prompts the user for the &LOWSTOCK variable value.

For interactive sessions, instruct users to enter EXIT on the command line when they are finished viewing the report. The initial procedure runs repeatedly until an EXIT command is issued. Thus, pressing the End function key from the report panel reruns the initial procedure; it does not display the QMF home panel.

Additionally, when you use the DSQSRUN parameter, ensure that the DSQEC\_RERUN\_IPROC global variable is set to 0 and that the current object is not the QMF home panel.

### Related concepts

[Passing variable values to an initial procedure](#)

When you supply the name of an initial procedure on the DSQSRUN parameter, you can also supply values for one or more variables contained in the procedure.

### Passing variable values to an initial procedure

When you supply the name of an initial procedure on the DSQSRUN parameter, you can also supply values for one or more variables contained in the procedure.

Follow these rules when you specify variables for the DSQSRUN parameter:

- Put parentheses around the variable parameter list.
- Precede the variable name with an ampersand, and ensure that the string is in a *variable\_name=value* format.
- Ensure that the combined total of characters for the procedure name and the variable parameter list is 98 characters or less.
- Separate the variable parameter specifications using a single comma, one or more blanks, or a combination of a comma and blanks.

The following table lists the number of ampersands required to use a variable in each environment.

Environment	Number of additional ampersands	Example
CICS	0	&variable=value
TSO without ISPF	0	&variable=value
TSO with ISPF	1	&&variable=value
TSO without ISPF using CLISTs	2	&&&variable=value
TSO with ISPF using CLISTs	3	&&&&variable=value

When you specify the name of an initial procedure, QMF issues a RUN PROC command that runs the procedure. When you use variables in your procedure, the values you supply for these variables must conform to the syntax used for passing variables on a RUN command.

For example, suppose you frequently need two pieces of information about employees in your organization. One piece of information is the name of the employee, and the other varies. You might define a query that returns values for the NAME column and uses a variable for the name of the other column. The following figure shows an example query and procedure. The figure also shows how to pass a value for the variable when you enter the DSQSRUN parameter, and shows the resulting RUN PROC command that QMF issues.

---

#### Query (JONES.QUERY2)

```
SELECT NAME, &COL FROM Q.STAFF
```

#### Procedure (JONES.PROC2)

```
RUN QUERY JONES.QUERY2 (&&COL=&COL
```

#### DSQSRUN parameter

```
QMFn I=JONES.PROC2(&COL=YEARS)
```

#### Resulting RUN command

```
RUN PROC JONES.PROC2 (&COL=YEARS)
```

Figure 25. Passing a column name on the DSQSRUN parameter

---

The next figure shows a similar example, but instead of passing one column name to the procedure, it allows you to pass several, which return the employee's name, the department, and the employee's salary.

---

**Query (JONES.QUERY3)**

```
SELECT &COLS FROM Q.STAFF
```

**Procedure (JONES.PROC3)**

```
RUN QUERY JONES.QUERY3 (&&COLS=&COLS
```

**DSQSRUN parameter**

```
QMFn I=JONES.PROC3(&COLS=((DEPT,NAME,SALARY))
```

**Resulting RUN command**

```
RUN PROC JONES.PROC3(&COLS=((DEPT,NAME,SALARY)))
```

*Figure 26. Passing several column names on the DSQSRUN parameter*

---

The next four examples show how to pass information that you normally supply after the WHERE keyword in a query.

These examples contain character strings, which require special syntax because of how QMF evaluates the values when it processes the RUN PROC command. Special characters (commas, blanks, parentheses, double or single quotation marks, apostrophes, and equal signs) can also be included in the string, as shown in the examples.

The first of these four examples shows a query that returns the names and employee numbers of all the managers in an organization. When you pass the character string MGR on the DSQSRUN parameter, be sure to enclose the value in single quotation marks.

---

**Query (JONES.QUERY4)**

```
SELECT JOB, NAME, ID FROM Q.STAFF WHERE JOB=&JOB
```

**Procedure (JONES.PROC4)**

```
RUN QUERY JONES.QUERY4 (&&JOB=&JOB
```

**DSQSRUN parameter**

```
QMFn I=JONES.PROC4(&JOB='MGR')
```

**Resulting RUN command**

```
RUN PROC JONES.PROC4 (&JOB='MGR')
```

*Figure 27. Passing a character string within single quotations marks on the DSQSRUN parameter*

---

The second of the four examples shows how to pass variable values that contain commas. Enclose the value SAN JOSE, CA in single quotation marks because it contains a comma.

---

**Query (JONES.QUERY5)**

```
SELECT * FROM Q.APPLICANT WHERE ADDRESS=&CITY
```

**Procedure (JONES.PROC5)**

```
RUN QUERY JONES.QUERY5 (&&CITY=&CITY
```

**DSQSRUN parameter**

```
QMFn I=JONES.PROC5(&CITY='SAN JOSE, CA')
```

**Resulting RUN command**

```
RUN PROC JONES.PROC5 (&CITY='SAN JOSE, CA')
```

*Figure 28. Passing a comma within a string on the DSQSRUN parameter*

---

The third of the four examples with character strings shows how to pass variable values that contain single quotation marks (for example, an apostrophe in a name). When you pass the value on the DSQSRUN parameter, be sure to enclose the value in single quotation marks and use two single quotation marks for the apostrophe instead of one.

---

**Query (JONES.QUERY6)**

```
SELECT * FROM Q.STAFF WHERE NAME=&NAME
```

**Procedure (JONES.PROC6)**

```
RUN QUERY JONES.QUERY6 (&&NAME=&NAME
```

**DSQSRUN parameter**

```
QMFn I=JONES.PROC6(&NAME='O' 'BRIEN')
```

**Resulting RUN command**

```
RUN PROC JONES.PROC6 (&NAME='O' 'BRIEN')
```

*Figure 29. Passing an apostrophe as part of a string on the DSQSRUN parameter*

---

The last of the four examples shows how to pass more than one variable value on the RUN command.

---

**Query (JONES.QUERY7)**

```
SELECT * FROM Q.STAFF WHERE DEPT IN &DEPT AND JOB = &JOB
```

**Procedure (JONES.PROC7)**

```
RUN JONES.QUERY7 (&&DEPT=&V1 &&JOB=&V2
```

**DSQSRUN parameter**

```
QMFn I=JONES.PROC7(&V1=((10,38)) &V2='MGR')
```

**Resulting RUN command**

```
RUN PROC JONES.PROC7(&V1=((10,38)) &V2='MGR')
```

*Figure 30. Passing multiple variable values on the DSQSRUN parameter*

---

**Related reference**

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

---

## Setting tracing options

---

QMF provides a trace facility that helps track QMF and user activity.

**About this task**

Use the DSQSDEBUG parameter to specify the level of detail at which you want to trace QMF activity. Specify a value of ALL on the DSQSDEBUG parameter to trace QMF activity at the highest level of detail, including program initialization errors and other errors that might occur before the user's profile is established. This value is shown in the following two different types of initialization commands. Replace *n* in the commands with the 1-character national language identifier for the language you are using.

```
DSQQMFn T=ALL
QMFn T=ALL
```

When you set DSQSDEBUG to NONE, the level of detail in the trace output depends on whether the QMF session is running interactively or in batch mode:

- In either an interactive or a batch session, only system error tracing is done during initialization, before the user's profile is established. The only way to turn off this initial tracing is to not allocate or define storage for the trace data.

- In a batch session, all messages and commands are traced at the most detailed level.

The trace level that you set with this parameter is effective until the user issues a SET PROFILE (TRACE=value command to change it or, in the case of a value of NONE, until the profile is loaded.

### Related concepts

[Getting the right level of detail in your trace output](#)

You can trace all QMF functions in detail or trace individual QMF functions.

### Related tasks

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Setting the trace for TSO

To set the trace in QMF for TSO, specify DSQSDEBUG=ALL when starting QMF.

Do not specify the DSQSDEBUG parameter when you start QMF for TSO as a Db2 for z/OS stored procedure. When QMF is started in this manner, you use the CALL statement to pass an input parameter that specifies the level of trace detail.

After QMF starts, you can turn tracing off if necessary by using the command SET PROFILE (TRACE=NONE. You can also set more specific levels of trace detail using the SET PROFILE command.

### Related concepts

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

[The trace facility](#)

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

### Related reference

[Summary of program parameters](#)

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## Setting the trace for CICS

You set several parameters to set the trace in QMF for CICS.

### Procedure

To specify tracing in QMF for CICS, follow these steps:

1. Specify DSQSDEBUG=ALL when starting QMF.
2. Use the DSQSDBQT parameter to specify the type of storage to be used for the trace data.
  - Specify the value TS to use a CICS auxiliary temporary storage queue for tracing:

```
QMFn DSQSDBQT=TS
```

- Use temporary storage for message-level tracing. Other types of tracing, such as ALL, capture more trace data and therefore require more storage. Use a transient data queue (a value of TD) if you think that the trace output could exceed the maximum size of a CICS temporary storage queue.

- A transient data queue named DSQD is predefined for you during QMF installation. If you use the DSQSDBQN parameter to name the transient data queue something other than DSQD, you must predefine the queue in the CICS DCT before you use it for the first time.
3. Use the DSQSDBQN parameter to specify the name of the CICS storage queue that will store the trace data.
- DSQSDBQN specifies the name of the transient data or temporary storage queue that holds trace data.
  - Ensure that the queue name conforms to CICS specifications for the type of queue specified by DSQSDBQT. TD queues have names from 1 to 4 characters. TS queues have names from 1 to 8 characters.
  - You do not need to predefine temporary storage queues to CICS. For example, the following statement dynamically allocates a temporary storage queue named MYTRACE to hold trace data for the QMF session:
- ```
QMFn DSQSDBQN=MYTRACE,DSQSDBQT=TS
```
- You must associate a transient data queue with a ddname. DSQDEBUG is the default name for the trace data set. QMF issues CICS ENQ and DEQ commands around single trace entries in the queue, so that a single queue can be used by more than one user.

## Results

After QMF starts, you can turn tracing off if necessary by using the command `SET PROFILE (TRACE=NONE`. You can also set more specific levels of trace detail using the `SET PROFILE` command.

## Related tasks

### The trace facility

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

## Related reference

### Summary of program parameters

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.

## Printing DBCS data from non-DBCS display devices

---

If you use the Uppercase, Japanese, or Korean National Language Feature (NLF), you might need to print double-byte character set (DBCS) data.

## Procedure

- To print DBCS data from non-DBCS display devices, set the DSQSDBCS program parameter to YES.

## Example

Suppose that a user you support uses a non-DBCS display device and needs to print a table (named DBCSTABLE) whose nonnumeric columns contain DBCS data. The following statement starts the Uppercase NLF from a cleared CICS screen and allows the user to print DBCSTABLE by using a command such as `PRINT DBCSTABLE (PRINTER=printername`.

```
QMFU K=YES
```

## Related concepts

### Setting up printing and charting functions

QMF end users frequently need to print data they retrieve from the database. This data might be in the format of a report, chart, database table, or some other QMF or database object.

**Related reference**

Summary of program parameters

A quick reference is provided for program parameters that you can pass to QMF when you start QMF under the standard interface.



---

# Chapter 11. Registering users and setting privileges

To enable users to access and use QMF, you must grant them access to the QMF application plan and packages and to database objects. You must also create a user profile. You can also customize users' object lists so that they can work with their objects more efficiently.

---

## Controlling access to the application plan and packages

To control access to QMF, you can grant and revoke access to the QMF application plan and packages as necessary.

### Granting access to the application plan and packages

The QMF application plan and packages enable QMF to run as a Db2 application program. To access QMF, all QMF users need access to the QMF application plan and packages.

At installation time, the EXECUTE privilege is granted to PUBLIC for the QMF plan (named QMF12 by default) and packages. You can revoke access on the QMF plan from PUBLIC. However, you then must issue an SQL GRANT statement for each individual QMF user, as in the following example.

```
GRANT EXECUTE ON PLAN qmflplanname TO userid
```

You can also restrict access to QMF through the QMF profile structure that you create.

#### Related concepts

[Revoking access to the application plan and packages](#)

To revoke a GRANT statement to the QMF application plan and packages, you must have SYSADM (or equivalent) authority.

[Establishing a profile structure for your site](#)

Users can use either a generic or a unique profile to access QMF. If necessary, you can restrict QMF access to those users who have unique profiles.

### Revoking access to the application plan and packages

To revoke a GRANT statement to the QMF application plan and packages, you must have SYSADM (or equivalent) authority.

To revoke a user's access to the QMF application plan, you can issue a statement like this one:

```
REVOKE EXECUTE ON PLAN qmflplanname FROM userid
```

The default QMF plan name for Version 12.1 is QMF12.

If it is possible that the user's EXECUTE privilege was granted by more than one user, use the following SQL statement to revoke the privilege:

```
REVOKE EXECUTE ON PLAN qmflplanname FROM userid BY ALL
```

To revoke these privileges from PUBLIC, substitute PUBLIC for *userid* in the statements.

If the user you are removing is a former QMF administrator who granted access to the QMF plan and packages to other users, removing access from the administrator also removes access for those users.

## Creating QMF user profiles

All QMF users need access to a user profile, which determines how QMF handles individual input from specific users. Use the profile to control certain aspects of a user's environment, such as where printer output is routed or whether input is converted to upper case.

### About the Q.PROFILES table

Each aspect of a user's QMF session maps to a value in a column of the Q.PROFILES control table. Each row of the Q.PROFILES table is an individual user profile.

The following table shows the columns of the Q.PROFILES control table. Each column of the table represents an aspect of a user's QMF session that you can customize. The defaults that are listed are for the English QMF environment.

Default values might be different for the English environment and some NLFs. For example, do not assume that the default for all NLFs is UPPER because the English default is UPPER. The default value for the CASE field in the German NLF is MIXED, and might also vary for other NLFs. For the default values for each NLF, see the translated versions of the Q.PROFILES table.

The Q.PROFILES table has the index Q.PROFILEX, with the attributes UNIQUE and CLUSTER. The keyed columns are CREATOR, TRANSLATION, and ENVIRONMENT. No three rows can have identical values for these three columns.

When you start QMF for TSO as a Db2 for z/OS stored procedure, the input parameters that you specify on the CALL statement that starts QMF override some of the settings in the QMF profile, as indicated in the table.

| Column name | Data type and length                                                                                                                | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREATOR     | VARCHAR(128)<br><b>Exception:</b> The CREATOR column is defined as CHAR(8) when you are connected to a DB2 for VSE and VM database. | No            | <b>Function</b><br>Specifies the user ID of the user who owns the profile.<br><b>Values</b><br>SYSTEM (default), Db2 primary or SQL authorization ID, or TSO logon ID (if the DSQSPRID parameter is set to TSOID).<br>The SYSTEM row is shipped with the Q.PROFILES table for English and each NLF; users who do not have unique profile rows can use the SYSTEM row.<br>When you start QMF for TSO as a Db2 for z/OS stored procedure, CREATOR must either be SYSTEM or the authorization ID that starts the WLM-managed address space from which the Q.DSQMFSP procedure runs. Any other values cause QMF initialization errors. |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------|----------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CASE        | CHAR(18)             | Yes           | <p><b>Function</b><br/>Specifies whether terminal input is converted to upper case.</p> <p><b>Values</b><br/>UPPER (default), STRING, or MIXED.</p> <p>Use a value of MIXED if you plan to take advantage of RACF support in TSO for mixed-case passwords or the CONNECT command will fail. In this case, instruct QMF users to enter all input in uppercase because QMF recognizes commands only in uppercase.</p> <p>For a full description of these values, see . CASE might have a different default for NLF users.</p> |
| DECOPT      | CHAR(18)             | Yes           | <p><b>Function</b><br/>Specifies the separators that QMF uses in numeric report columns.</p> <p><b>Values</b><br/>PERIOD (default), COMMA, and FRENCH. For more information, see . DECOPT is translated and might have a different default for NLF users.</p>                                                                                                                                                                                                                                                               |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|----------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONFIRM     | CHAR(18)             | Yes           | <p><b>Function</b><br/>Specifies the default action for confirmation prompting with QMF commands that support the CONFIRM option. This default applies when the commands do not specify the CONFIRM option. The value of CONFIRM indicates whether QMF displays a confirmation panel before completing a command that:</p> <ul style="list-style-type: none"> <li>• Changes or replaces an object in the database</li> <li>• Has a cost estimate that exceeds allocated resources defined in the Db2 resource limit facility (the Db2 governor)</li> <li>• Replaces a file or data set (in the case of an EXPORT command, for example)</li> </ul> <p><b>Values</b></p> <p><b>YES (default)</b><br/>Displays a confirmation panel, providing an opportunity to cancel the command before it runs</p> <p>A value of YES is ignored when QMF for TSO has been started as a Db2 for z/OS stored procedure.</p> <p><b>NO</b><br/>Completes the command without displaying a confirmation panel</p> <p>When you run a query that contains multiple SQL statements that change the database, a single confirmation panel is displayed. The answer that you provide in response to this prompt applies to changes that will be made by all SQL statements in the query.</p> |
| WIDTH       | CHAR(18)             | Yes           | <p><b>Function</b><br/>Controls the number of printed columns per page.</p> <p><b>Values</b><br/>22 to 999. The default is 132.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| LENGTH      | CHAR(18)             | Yes           | <p><b>Function</b><br/>Controls the number of printed lines per page.</p> <p><b>Values</b><br/>1 to 999, or CONT if you want no page breaks. The default is 60.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length | Nulls allowed | Function and possible values                                                                                                                                                                                                             |
|-------------|----------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LANGUAGE    | CHAR(18)             | Yes           | <p><b>Function</b><br/>Controls which query language QMF uses when creating a new query after a RESET QUERY command is issued.</p> <p><b>Values</b><br/>SQL (default), QBE (for Query-by-Example), or PROMPTED (for Prompted Query).</p> |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|----------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPACE       | CHAR(50)             | Yes           | <p><b>Function</b><br/>Specifies the default storage space in the database for tables created with the SAVE DATA, IMPORT, or RUN QUERY command.</p> <p><b>Values</b><br/>The name of a valid storage space for the current database location. Depending on the database type, this could be a database name, a table space name, or a combination of a database and table space name. You can specify the following storage locations:</p> <p>You can specify the following storage locations for Values:</p> <p><b>databaseName.spacename</b><br/>Specify both a database and a table space to have QMF save the table in that specific table space.</p> <p>For example, this query configures the profile of a QMF user named SMITH to save tables in table space TSPACE1, in database DBASE1:</p> <pre data-bbox="878 1024 1471 1136">UPDATE Q.PROFILES SET SPACE='DBASE1.TSPACE1' WHERE CREATOR='SMITH' AND TRANSLATION='ENGLISH'</pre> <p>However, if all users are using the same table space, you could experience resource contention.</p> <p><b>DATABASE "databaseName"</b><br/>When you specify the DATABASE keyword followed by a database name in quotes, each table is created in a separate table space created implicitly and exclusively for that table by Db2. Such implicit table spaces have default LOCKSIZE, BUFFERPOOL, STOGROUP, and space attributes, and have names that match the name of the table that was created.</p> <p>For example, the following value for the SPACE field of the Q.PROFILES table saves each table in a separate table space in the DSQDBDEF database:</p> <pre data-bbox="878 1682 1471 1734">DATABASE "DSQDBDEF"</pre> <p>Limiting each table space to one table only minimizes resource contention. Both partitioned and universal table space schemes impose a limit of one table per table space. QMF recommends this method.</p> <p><b>NULL or blank</b><br/>If the SPACE field of the QMF profile is null or blank, SAVE DATA or IMPORT TABLE commands create tables in QMF for IS created table space names in the DSNDB04 database by default.</p> |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|----------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TRACE       | CHAR(18)             | Yes           | <p><b>Function</b><br/>Controls the level of detail in trace output.</p> <p><b>Values</b><br/>ALL traces all functions at the most detailed level. A character string of function codes and numbers indicates the level of tracing for individual QMF functions. The default value varies depending on the value provided for the DSQSMODE parameter. For example, when the DSQSMODE parameter is set to B, the trace level is L2; otherwise it is NONE.</p> <p>Only the values ALL and NONE are translated in NLFs.</p> <p>When you start QMF for TSO as a Db2 for z/OS stored procedure, values for the trace options are taken from the <b>trace-level</b> parameter of the CALL statement that starts QMF. You can include a SET PROFILE (TRACE command in the initial procedure specified by the <b>object-name</b> parameter to change the level of trace detail for the duration of the stored procedure session, as long as trace output is set to go to the DSQDEBUG data set. If <b>trace-level</b> is set to L2, <i>L2-destination</i> must be set to DSQDEBUG for SET PROFILE (TRACE commands to be accepted.</p> <p>For more information, see <a href="#">“Getting the right level of detail in your trace output”</a> on page 358 and <a href="#">“Tracing individual QMF modules”</a> on page 359</p> |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length                                                                                                                    | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRINTER     | CHAR(8)                                                                                                                                 | Yes           | <p><b>Function</b><br/>Controls where printer output is routed.</p> <p><b>Values</b><br/>Use a null (default) or blank value to route print output to CICS temporary storage or transient data queues, or to the data set with the ddname DSQPRINT. Use a GDDM nickname to direct output to a printer defined by GDDM.</p> <p>If you allocate output from DSQPRINT to go to the HOLD queue, to release the output to the OUTPUT queue for printing you must issue the following TSO command:</p> <pre>FREE DDNAME(DSQPRINT)</pre> <p>When you start QMF for TSO as a Db2 for z/OS stored procedure and want to receive report output back in result sets, this profile option must be set to a string of blanks or a string of blanks must be passed on a SET PROFILE command in the initial procedure that runs after QMF starts.</p> |
| TRANSLATION | CHAR(18)                                                                                                                                | No            | <p><b>Function</b><br/>Indicates whether the language in use is English or a National Language Feature (NLF).</p> <p><b>Values</b><br/>English (default) or the name that QMF uses for the NLF.</p> <p>When you start QMF for TSO as a Db2 for z/OS stored procedure, the language value is taken from the <b>language</b> parameter passed on the CALL statement that starts QMF and not from the profile.</p> <p>For information about NLFs, see <a href="#">Table 27 on page 102</a>.</p>                                                                                                                                                                                                                                                                                                                                           |
| PFKEYS      | VARCHAR(261)<br><br><b>Exception:</b> The PFKEYS column is defined as CHAR(31) when you are connected to a DB2 for VSE and VM database. | Yes           | <p><b>Function</b><br/>Indicates the table or view (if any) where users' customized function key definitions are stored.</p> <p><b>Values</b><br/>Any valid Db2 table or view name. If this value is blank or null (the default), the default keys are used.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |



Table 36. Structure of the Q.PROFILES table (continued)

| Column name    | Data type and length                                                                                                                      | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SYNONYMS       | VARCHAR(261)<br><br><b>Exception:</b> The SYNONYMS column is defined as CHAR(31) when you are connected to a DB2 for VSE and VM database. | Yes           | <p><b>Function</b><br/>Indicates the table or view where users' customized command definitions are stored. When a user enters a command synonym, the synonym definition must be stored in the table named here or the command fails.</p> <p><b>Values</b><br/>Any valid Db2 table or view name.</p> <p>The table named here must conform to the structure explained in “Customizing Command synonyms” on page 234. You can use the default synonyms, create your own table of synonyms, or create your own table and add the synonyms supplied by QMF to it.</p> <p>Default QMF command synonyms are stored in the Q.COMMAND_SYNONYMS table. For NLF users, default synonyms are stored in the Q.COMMAND_SYNONYM_n table, where n is a 1-character language identifier from Table 27 on page 102.</p> <p>If this value is blank or null (the default), no customized definitions are used.</p> |
| RESOURCE_GROUP | CHAR(16)                                                                                                                                  | Yes           | <p><b>Function</b><br/>Controls how the default governor exit routine limits a user's resources or commands.</p> <p><b>Values</b><br/>Any valid resource group name. If this value is blank or null (the default), QMF attempts to use the user's primary or SQL authorization ID here, according to the value of the DSQSPRID program parameter: If DSQSPRID is set to TSOID, QMF attempts to assign the TSO logon ID as the value for the RESOURCE_GROUP column; if DSQSPRID is set to PRIMEID, QMF attempts to assign the Db2 or SQL primary authorization ID as the value of the RESOURCE_GROUP column. In these cases, the user's session is not governed (unless the TSO ID or the primary or SQL authorization ID is a valid resource group name).</p> <p>The resource group in use for the user's QMF session is recorded in the DSQAP_RESOURC_GRP global variable.</p>                |

Table 36. Structure of the Q.PROFILES table (continued)

| Column name | Data type and length | Nulls allowed | Function and possible values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------|----------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MODEL       | CHAR(8)              | Yes           | <p><b>Function</b><br/>Specifies the model for data access.</p> <p><b>Values</b><br/>Always use the value REL for this column, indicating relational data.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| ENVIRONMENT | CHAR(8)              | Yes           | <p><b>Function</b><br/>Indicates the operating environment.</p> <p><b>Values</b><br/>Valid values for new installations are TSO or CICS. The default value is TSO. If the Q.PROFILES table has been migrated from a prior release, you might see the following values in this column in addition to TSO or CICS.</p> <p><b>CMS</b><br/>Used in QMF Version 7.2 and earlier</p> <p><b>CICSMVS</b><br/>Used in QMF Version 7.2 and earlier to distinguish QMF installations under z/OS from those under VSE</p> <p><b>CICSVSE</b><br/>Used in QMF Version 7.2 and earlier to distinguish QMF installations under VSE from those under z/OS</p> <p><b>WINDOWS</b><br/>Used in QMF for Workstation and QMF for WebSphere®</p> |

### Related concepts

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## Establishing a profile structure for your site

Users can use either a generic or a unique profile to access QMF. If necessary, you can restrict QMF access to those users who have unique profiles.

### Generic profiles

You can allow users to use the default QMF profile, which is the row of the Q.PROFILES table where the CREATOR column has a value of SYSTEM.

The Q.PROFILES table is shipped with default profile values in this row. You can change the default values to create a generic profile that meets the needs of your site.

## Unique profiles

If you choose not to maintain a generic QMF profile, you must create a unique row in the Q.PROFILES table for each user. Set the CREATOR column of Q.PROFILES to the primary authorization ID of the user and customize other column values according to individual needs. If you start QMF in TSO with a DSQSPRID value of TSOID, the CREATOR column is the user's TSO logon ID.

You can create unique profiles for some users at your site and allow other users to use the generic profile.

## Restricting QMF usage to those with unique profiles

It can be difficult to track individual resource use if several people use QMF under the common, default SYSTEM profile. To restrict use of QMF to users who have unique profiles, delete the SYSTEM rows of Q.PROFILES. The following table shows SQL statements that delete the rows. A statement that deletes the SYSTEM row from the German version of the Q.PROFILES table is provided as an example of how to delete the row for an NLF. For the value of the TRANSLATION column, use the name that QMF uses for the NLF.

**Important:** For both English and NLF environments, always specify a TRANSLATION value when deleting rows from Q.PROFILES, or more rows (across different national language environments) might be deleted than you intend. Additionally, always use a WHERE clause, or all rows of Q.PROFILES are deleted.

| Base QMF (English)                                                                 | German NLF                                                                         |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| <pre>DELETE FROM Q.PROFILES WHERE CREATOR='SYSTEM' AND TRANSLATION='ENGLISH'</pre> | <pre>DELETE FROM Q.PROFILES WHERE CREATOR='SYSTEM' AND TRANSLATION='DEUTSCH'</pre> |

After you delete the SYSTEM row of Q.PROFILES, you must create a unique profile for every QMF user; otherwise, your users will not be able to use QMF.

## Related tasks

### [Adding a user profile](#)

Add a row in the Q.PROFILES table for each unique user profile you create.

### [Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Related reference

### [About the Q.PROFILES table](#)

Each aspect of a user's QMF session maps to a value in a column of the Q.PROFILES control table. Each row of the Q.PROFILES table is an individual user profile.

## Adding a user profile

Add a row in the Q.PROFILES table for each unique user profile you create.

## About this task

When QMF is started, it determines which users are authorized to establish a QMF session by searching the CREATOR, ENVIRONMENT, and TRANSLATION columns of the Q.PROFILES table. QMF searches for specific profile values in the following order:

1. CREATOR=*userid*, ENVIRONMENT=*current operating environment*
2. If running in CICS, CREATOR=*userid*, ENVIRONMENT=CICS
3. CREATOR=*userid*, ENVIRONMENT=NULL
4. CREATOR=SYSTEM, ENVIRONMENT=*current operating environment*

5. If running in CICS, CREATOR=SYSTEM, ENVIRONMENT=CICS

6. CREATOR=SYSTEM, ENVIRONMENT=NULL

QMF must find values for CREATOR and ENVIRONMENT that match one of the pairs in the preceding list, or QMF initialization ends in an error before the QMF home panel is displayed. If the values in the CREATOR column of your Q.PROFILES table are TSO logon IDs instead of Db2 primary authorization IDs, set the DSQSPRID parameter to TSOID when you start QMF; otherwise authentication with the Q.PROFILES table fails when users attempt to log on to QMF.

### Procedure

To add a user profile to the Q.PROFILES table, follow these steps:

1. Write an SQL INSERT statement to add the row.

**Tip:** To make maintenance tasks more efficient, a single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command. Alternatively, you can create a template query that describes a standard profile and use a substitution variable for any value that will change from one profile to another (such as the value for the CREATOR column).

2. At a minimum, specify values for the CREATOR, ENVIRONMENT, and TRANSLATION columns of the table:

- CREATOR

Specify a value of SYSTEM if you are creating a generic QMF profile. If you are creating a unique profile, specify the authorization ID of the user. The database uses this ID to determine if the user is authorized to use the database.

- ENVIRONMENT

Specify a value of CICS or TSO. A value of TSO includes TSO, ISPF, or native z/OS.

- TRANSLATION

Specify ENGLISH or the translated value for the NLF you are using. For example, profile rows for French users contain FRANCAIS in the TRANSLATION column.

Always specify a TRANSLATION value when inserting a row into Q.PROFILES, or the TRANSLATION value defaults to a null value and the profile row is automatically ignored.

You can establish multiple NLF profiles for the same user. For example, a user can have a profile with one set of values in one national language and a profile with a different set of values in another national language.

3. Issue the RUN QUERY command to add the profile rows.

### Example

This table shows sample SQL that creates unique profiles in the TSO environment for users with SQL authorization IDs of JONES (base QMF, or English) and SCHMIDT (German NLF).

| Base QMF (English)                                                                                                                                                                                                        | German NLF                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>INSERT INTO Q.PROFILES (CREATOR, LANGUAGE, SPACE, TRANSLATION, PFKEYS, SYNONYMS, RESOURCE_GROUP, ENVIRONMENT) VALUES ('JONES', 'PROMPTED', 'SAVEIT' 'ENGLISH', 'PFKEYS', 'COMMAND_SYNONYMS' 'NONPRIME', 'TSO')</pre> | <pre>INSERT INTO Q.PROFILES (CREATOR, LANGUAGE, SPACE, TRANSLATION, PFKEYS, SYNONYMS, RESOURCE_GROUP, ENVIRONMENT) VALUES ('SCHMIDT', 'MENEUE', 'STUT2BER' 'DEUTSCH', 'DEUTASTEN', 'COMMAND_SYNONYM_D', 'SCHICHT', 'TSO')</pre> |

## Related tasks

### [Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Related reference

### [About the Q.PROFILES table](#)

Each aspect of a user's QMF session maps to a value in a column of the Q.PROFILES control table. Each row of the Q.PROFILES table is an individual user profile.

## Updating a user profile

You can change the values in a user's profile by using either the SET PROFILE command or SQL UPDATE statements.

### Using the SET PROFILE command

Using the SET PROFILE is quicker than using SQL UPDATE statements, because you can enter it from the QMF command line with minimal typing.

Values set using SET PROFILE remain effective only until the user's session ends; use the SAVE PROFILE command to save values you changed.

Because no special SQL privileges are required to use this command, your users can easily update their own profiles. However, they cannot use SET PROFILE to update fields that you might use to customize their QMF sessions. These fields are PFKEYS, SYNONYMS, and RESOURCE\_GROUP. You can use SQL UPDATE statements or the QMF Table Editor to update these Q.PROFILES fields.

## Related concepts

### Using SQL UPDATE statements

SQL UPDATE statements can be used to update all fields of the Q.PROFILES table, including SYNONYMS, PFKEYS, and RESOURCE\_GROUP.

### Updating the SYSTEM profile

You can change the default values provided in the SYSTEM row of the Q.PROFILES table. However, any user who needs different values than those you assigned for the SYSTEM row must have a unique profile row.

### Using SQL UPDATE statements

SQL UPDATE statements can be used to update all fields of the Q.PROFILES table, including SYNONYMS, PFKEYS, and RESOURCE\_GROUP.

**Tip:** To make maintenance tasks more efficient, a single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command.

Use an SQL UPDATE statement similar to the one in the following table to update existing user profiles. This example changes the name of the table that stores a user's command synonyms. On the left is an example statement for user JONES for base QMF (English); on the right is the same statement for user SCHMIDT for the German NLF.

| <b>Base QMF (English)</b>                                                                                    | <b>German NLF</b>                                                                                        |
|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <pre>UPDATE Q.PROFILES SET SYNONYMS='COMMAND_SYNONYMS' WHERE CREATOR='JONES' AND TRANSLATION='ENGLISH'</pre> | <pre>UPDATE Q.PROFILES SET SYNONYMS='GUMMOW.XYZ' WHERE CREATOR='SCHMIDT' AND TRANSLATION='DEUTSCH'</pre> |

**Important:** When running UPDATE, DELETE, and INSERT statements on the Q.PROFILES table, always include the TRANSLATION column in the statement; otherwise, QMF applies the changes to all language environments.

**Related concepts**

Using the SET PROFILE command

Using the SET PROFILE is quicker than using SQL UPDATE statements, because you can enter it from the QMF command line with minimal typing.

Updating the SYSTEM profile

You can change the default values provided in the SYSTEM row of the Q.PROFILES table. However, any user who needs different values than those you assigned for the SYSTEM row must have a unique profile row.

**Updating the SYSTEM profile**

You can change the default values provided in the SYSTEM row of the Q.PROFILES table. However, any user who needs different values than those you assigned for the SYSTEM row must have a unique profile row.

For example, suppose that your site has two groups of users: the PRIME group contains the majority of QMF users, who all use the system during normal business hours; the NONPRIME group contains fewer users, who all use the system during non-business hours. Suppose that PRIME is the default value for the RESOURCE\_GROUP field of the SYSTEM row in Q.PROFILES. You must formally enroll the users who are in the NONPRIME group by giving them unique profile rows.

**Related concepts**

Using the SET PROFILE command

Using the SET PROFILE is quicker than using SQL UPDATE statements, because you can enter it from the QMF command line with minimal typing.

Using SQL UPDATE statements

SQL UPDATE statements can be used to update all fields of the Q.PROFILES table, including SYNONYMS, PFKEYS, and RESOURCE\_GROUP.

**Deleting a user profile**

Periodically, you might need to delete obsolete user profiles from the Q.PROFILES table. Delete a user profile from Q.PROFILES only when you are sure that objects created by the primary authorization ID or the TSO logon ID associated with the profile are either deleted or transferred to other users.

Use a DELETE statement similar to the following one to delete a user profile. You can include several DELETE statements in one SQL query to speed up maintenance tasks.

| <i>Table 40. Deleting a QMF user profile</i>                                      |                                                                                     |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <b>Base QMF (English)</b>                                                         | <b>German NLF</b>                                                                   |
| <pre>DELETE FROM Q.PROFILES WHERE CREATOR='JONES' AND TRANSLATION='ENGLISH'</pre> | <pre>DELETE FROM Q.PROFILES WHERE CREATOR='SCHMIDT' AND TRANSLATION='DEUTSCH'</pre> |

**Important:** Make sure that each DELETE statement specifies a value for the TRANSLATION column if you want to delete the user profile in a single NLF environment. If you do not specify a value for the TRANSLATION column, QMF deletes the profile for all languages.

If the user whose profile you deleted had a private table space, use the DROP TABLESPACE statement from the **SQL Query** panel if the space contains nothing you want to save. Also, you can use the SQL DROP TABLE statement or QMF ERASE commands if you want to delete specific QMF or database objects.

## Providing access to QMF and database objects

QMF objects, such as queries and procedures, and functions such as the Table Editor, allow users to access and manipulate data stored in tables in the database.

### About this task

To enable users to work with the QMF and database objects that they need to access, you must grant access to them. Because this data might be sensitive, you might need to control users' access to certain objects.

### Privileges required for QMF commands and functions

The process of controlling users' access to certain objects is started by determining the tasks that each user needs to perform. Then review the SQL privileges required for users to run QMF queries and commands and perform QMF functions.

For more information about privileges required to use QMF, review the following sections :

- [“SQL privileges required for QMF commands” on page 185](#)
- [“SQL privileges required for prompted and QBE queries” on page 186](#)
- [“SQL privileges required for the Table Editor” on page 186](#)

### SQL privileges required for QMF commands

Using the following table, locate the QMF command that your users require and grant them the required SQL privilege on the table or view they are working with.

| QMF command                                     | SQL privilege required on objects referenced by the command                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DISPLAY <i>tablename/</i><br><i>viewname</i>    | SELECT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| DRAW <i>tablename/</i><br><i>viewname</i>       | SELECT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| EDIT TABLE <i>tablename/</i><br><i>viewname</i> | The necessary privileges depend on the Table Editor mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| EXPORT TABLE<br><i>tablename/viewname</i>       | SELECT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| IMPORT TABLE<br><i>tablename/viewname</i>       | <p>If the table exists, SELECT, DELETE, and INSERT privileges are required. To include a comment, you must have either ownership of the table or DBADM (or equivalent) authority for the database in which the table resides. If the table does not exist, the user must have DBADM or equivalent authority for the database, the USE privilege for the table space specified in the SPACE field of the profile, or the CREATETAB privilege for the database in which the table will be created.</p> <p>Use the IMPORT command sparingly in CICS, because it can affect QMF performance for other users in the same address space. QMF uses GET/PUT services when operating under QSAM, which can lock out other QMF users in the same CICS region during I/O operations.</p> |
| PRINT <i>tablename/</i><br><i>viewname</i>      | SELECT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| RUN query                                       | Whatever privileges are used in the query                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Table 41. QMF commands and their SQL equivalents (continued)

| QMF command                               | SQL privilege required on objects referenced by the command                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RUN procedure                             | Whatever privileges are used in the commands in the procedure                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| SAVE DATA                                 | <p>If the table exists, SELECT, DELETE, and INSERT privileges for the table are required.</p> <p>If the table does not exist, you must have either the CREATETAB privilege or DBADM (or equivalent) authority for the database or the USE privilege for the table space specified in the SPACE field of the associated user profile.</p> <p>To include a comment, you must have either ownership of the table or DBADM (or equivalent) authority for the database where the table resides.</p> |
| LIST <i>tablename/</i><br><i>viewname</i> | SELECT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

### SQL privileges required for prompted and QBE queries

Using the following table, locate the type of query your users need and grant them the SQL privilege on the table or view against which the query runs.

Table 42. QMF query types and their SQL equivalents

| Query type | SQL privilege needed for the query |
|------------|------------------------------------|
| Prompted   | SELECT                             |
| QBE I.     | INSERT                             |
| QBE P.     | SELECT                             |
| QBE U.     | UPDATE                             |
| QBE D.     | DELETE                             |

### SQL privileges required for the Table Editor

Using the following table, locate the Table Editor function your users need to use and grant them the SQL privilege on the table or view that they need to edit.

Table 43. Table Editor commands and their SQL equivalents

| Table Editor function | SQL privilege needed on tables or views being edited |
|-----------------------|------------------------------------------------------|
| ADD                   | INSERT                                               |
| SEARCH                | SELECT                                               |
| CHANGE                | UPDATE                                               |
| DELETE                | DELETE                                               |

### Related concepts

[Granting a user the privileges to create tables](#)



The authority that you must grant to a user who creates tables varies. One set of privileges are required when you specify a database and table space name in the SPACE field of the QMF user profile. A different set of privileges is required when you specify only a database or no value.

## Granting and revoking privileges

Users own objects that they create and save in the database (unless they create a table with a different owner). The object owner can grant or revoke SQL privileges on these objects to other users.

Anyone with DBADM or equivalent authority can grant or revoke SQL privileges for any object in the database for which they hold this authority. If you do not have DBADM or equivalent authority and you do not own the object, you must have the grant option for the privileges you hold on that object. (The grant option is specified by using the WITH GRANT OPTION clause on the GRANT statement.) When granting or revoking privileges on objects you do not own, qualify the object with the SQL authorization ID of the owner:

```
JONES.ORDER_BACKLOG
```

SQL authorization IDs can be implicit qualifiers. Queries can contain unqualified table, view, and index names. QMF commands can contain unqualified query, procedure, form, and analytics objects names. In these cases, the user's SQL authorization ID serves as the implicit qualifier. For example, suppose that a user is operating with JONES as the current SQL authorization ID. During the QMF session, the user issues the following command:

```
RUN QUERYA (FORM=FORMA
```

This command runs the following SQL query:

```
SELECT * FROM TABLEA
```

The RUN command refers to the query JONES.QUERYA and the form refers to the form JONES.FORMA. The SELECT command refers to the table JONES.TABLEA.

If you attempt to create a table, view, index, or alias with an unqualified name, your current authorization ID becomes the owner of the object. That ID must have the privileges needed to create the object.

You must have DBADM or equivalent authority to create a table, view, or index with a qualified name that is not your authorization ID.

### The SQL GRANT statement

You can use the SQL GRANT statement to grant SQL SELECT, UPDATE, INSERT, DELETE, and other privileges on tables or views.

For example, suppose user JONES needs to use the Change mode of the Table Editor for a table called ORDER\_BACKLOG. To grant JONES the UPDATE privilege on the ORDER\_BACKLOG table, issue the following statement:

```
GRANT UPDATE ON ORDER_BACKLOG TO JONES WITH GRANT OPTION
```

The WITH GRANT OPTION clause indicates that JONES can grant to other users any of the SQL privileges you granted for the ORDER\_BACKLOG table.

Use the keyword PUBLIC to grant SQL privileges to all local QMF users. For example, use the following statement to grant the INSERT privilege on the ORDER\_BACKLOG table to all users in the local database, and allow each of those users to grant the INSERT privilege to other users:

```
GRANT INSERT ON ORDER_BACKLOG TO PUBLIC WITH GRANT OPTION
```

To make an object available to local and remote users for Db2 for z/OS subsystems that have distributed data that is enabled, grant authority to PUBLIC AT ALL LOCATIONS. For example, the following statements give the SELECT privilege on the table Q.STAFF:

```
GRANT SELECT ON TABLE Q.STAFF TO PUBLIC
GRANT SELECT ON TABLE Q.STAFF TO PUBLIC AT ALL LOCATIONS
```

Q.STAFF is one of the QMF sample tables. Similar statements are run for all of the QMF sample tables during QMF installation so that all users have the SELECT privilege on the sample tables.

**Tip:** To make maintenance tasks more efficient, a single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command. Alternatively, you can create a template query that uses QMF variables in place of the parts of the query that frequently change (such as the type of privilege, the object name, and the authorization ID). You might also consider using a QMF procedure to do the task if there is more than one query.

### The SQL REVOKE statement

Issue REVOKE statements to withdraw privileges.

For example, the following statement withdraws the SELECT privilege from user BAKER on the table SMITH.TABLEA:

```
REVOKE SELECT ON TABLE SMITH.TABLEA FROM BAKER
```

You can always withdraw grants for which your authorization ID is the grantor.

If you revoke a privilege from a grantee and find that the grantee still has the privilege, that grantee received the privilege from another user. If your SQL authorization ID has SYSADM or equivalent authority, however, you can revoke the grants of others. By using SYSADM or equivalent authority you can revoke privileges even if they are a result of multiple grants. For example, BAKER has the SELECT privilege on the table SMITH.TABLEA. The QMF administrator wants to remove this privilege from BAKER, but does not know who the grantors are. The QMF administrator, who has SYSADM authority, can run the following statement:

```
REVOKE SELECT ON TABLE SMITH.TABLEA FROM BAKER BY ALL
```

The clause BY ALL removes every grant of the privilege.

Use the PUBLIC keyword to revoke privileges from all QMF users.

You cannot remove a table privilege from the owner of a table. Additionally, you cannot remove an implied database privilege, such as CREATETAB, from someone with, for example, DBADM authority over a database.

Database privileges have a cascading structure. Privileges that are revoked from a user are automatically revoked from any additional users to whom that user granted them.

The loss of privileges can spread to many users, especially if some of those users who lost privileges granted privileges to others. With this loss of privileges might come other losses as well:

- The owner of a view loses the view if the owner loses the SELECT privilege on one of the underlying objects. Views for which the lost view is an underlying object are also lost, and so on.
- A Db2 application plan can become invalid if the authorization ID under which it was bound loses a privilege that the plan needs for the operation of the program. For example, if the SELECT privilege is lost on a table, no one can run the program.

Problems that result from cascading privileges are more likely when many users can grant database privileges. So consider carefully which users in your organization you want to be responsible for this task.

**Tip:** If you must revoke different privileges from many users at once, you can include multiple REVOKE statements in a single SQL query. To create a query that includes multiple statements, place a semicolon

between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command. Alternatively, you can create a template query that uses QMF variables in place of the parts of the query that frequently change (such as the type of privilege, the object name, and the authorization ID). You might also use a QMF procedure to do the task if there is more than one query.

## Setting standards for objects and allowing uncommitted read

---

QMF objects might be shared among many users, so the objects should have names that indicate what the object is and how it should be used. Allowing uncommitted reads can be useful in a distributed environment.

### Sharing QMF objects with other users

You can enable other QMF users to access QMF objects that you own.

You can enable access to QMF queries, forms, and procedures by using the SHARE parameter of the QMF SAVE command. In QMF Analytics for TSO, you can enable access to your chart or statistics specifications by using the SHARE option that is available through the Save function key. Charts and statistics specifications are saved as analytics objects. All QMF folder objects contain information about the actual QMF object (query, procedure, form, or analytics object); the actual QMF object identifies the owner and the SHARE option for that object.

Encourage users to use standard names and to provide comments that describe for other users the purpose of queries, forms, procedures, tables, and analytics objects. Tables and views require more maintenance and administration, so consider establishing special guidelines for creating these objects.

Specify SHARE=YES when you save an object to allow any other user to display the query and use it in a QMF command that does not replace or erase it. For example, the following command saves the current query as ORDER\_QUERY and allows any other user to display and run it:

---

```
SAVE QUERY AS ORDER_QUERY (SHARE=YES)
```

*Figure 31. Sharing a QMF object*

---

Note that to save an ANALYTIC object, you must use the Save function key in QMF Analytics for TSO.

The default is defined by the global variable DSQEC\_SHARE.

The owner of an object can change its shared status at any time by using a DISPLAY command followed by a SAVE command, as shown here:

---

```
DISPLAY ORDER_QUERY  
SAVE QUERY AS ORDER_QUERY (SHARE=NO)
```

*Figure 32. Changing the shared status of a QMF object*

---

### Allowing uncommitted read

If you want your QMF session to allow uncommitted read, you can specify a value for the global variable DSQEC\_ISOLATION in the Q.SYSTEM\_INI procedure.

Values can be:

'0'

Isolation level UR (uncommitted read)

Uncommitted read can be useful in a distributed environment. However, if you are using uncommitted read, any reports that users view might contain data that was deleted from the database after the report was displayed.

#### '1'

Isolation level CS (cursor stability)

This is the default. When using cursor stability, QMF does not display the report until all database commands affecting the data in the report have been completed.

## Users' object lists

---

QMF users periodically need to list tables and views that they have saved in the database or view comments that show them what purpose a table or view serves or what type of data a column in the table or view contains. The QMF LIST and DESCRIBE commands perform these functions.

### Customizing users' object lists

Using the default views supplied by QMF for your table lists and column information can increase processing time, because Db2 gathers authorization information from the SYSIBM.SYSTABAUTH table. If you do not need the extra security provided by these authorization checks, consider creating your own views that generate a list of objects stored in the database.

#### About this task

**Note:** The installation option, EXTSEC, defined in installation exec QMF1210.SDSQEXCE(DSQ1DEFS), can influence the definition of the default views defined in installation job QMF1210.SDSQSAPE(DSQ1BVW). When set, the default views can be defined without any SYSIBM.SYSTABAUTH dependency.

To customize users' object lists, you create your own view, then run a SET GLOBAL command to use the view.

#### What to do next

In addition to customizing the default object list function, you can install an enhancement to the LIST command that lists tables and views authorized to either a primary authorization ID or a secondary authorization ID. This enhancement eliminates the need to grant privileges on these objects to PUBLIC if you want to see these objects in list results.

#### Related tasks

[Installing the enhanced LIST command function \(z/OS only\)](#)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.

#### Creating your own list view

You create your own view for the LIST or DESCRIBE commands by using a CREATE statement.

This statement eliminates duplicate rows in the view and, although Db2 spends more time before returning rows to QMF, there is less data transfer between the database and the user machine, producing better performance. You can name your customized view any name that is valid in QMF.

```

CREATE VIEW Q.DATABASE_OBJECTS
  (OWNER,TNAME,TYPE,SUBTYPE,MODEL, RESTRICTED, REMARKS,
   CREATED,MODIFIED, LAST_USED, LABEL, LOCATION, OWNER_AT_LOCATION,
   NAME_AT_LOCATION)
AS SELECT CREATOR,TNAME,
'TABLE',TABLETYPE,' ',' ',REMARKS,
' ',' ',' ',TLABEL,' ',' ',' '
FROM SYSIBM.SYSTABLES
  WHERE TNAME IN (SELECT TTNAME
                  FROM SYSIBM.SYSTABAUTH
                  WHERE TCREATOR = A.CREATOR
                     AND GRANTEETYPE = ' &'
                     AND GRANTEE IN (USER, 'PUBLIC'))

```

Figure 33. Customizing your object lists using global variables

**Tip:** To make maintenance tasks more efficient, a single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command.

Follow these rules if you are creating a list view of your own:

- The view must have the same column names as the corresponding view supplied by QMF. The column names in the CREATE VIEW statement of the alternative view can be in any order.
- All columns must have a data type of CHAR or VARCHAR. QMF returns errors upon finding other data types.
- Always supply values for OWNER, TNAME, TYPE, and CNAME. These columns cannot be null.

If you want to create a view that shows only the tables for which a user has privileges, but does not require a join, consider defining a view that selects only from SYSIBM.SYSTABAUTH, but does not return values for REMARKS or LABEL.

For other administrators, consider creating another view similar to the default QMF view, but that selects only from SYSIBM.SYSTABLES or SYSIBM.SYSCOLUMNS (for column information). Then the administrators can name this view in the DSQEC\_COLS\_LDB2 or DSQEC\_COLS\_RDB2 global variables and access descriptive information for any columns in the database.

### **Making your new view the default view**

To use a view that you have created instead of the default view, use a SET GLOBAL command to set the appropriate global variable to the new view name.

For example:

```
SET GLOBAL (DSQEC_TABS_LDB2 = QMFADM.LOCAL_DB2_TABLES
```

### **Related concepts**

[Global variables that store the default view names](#)

The names of the default views used in LIST and DESCRIBE commands are stored in QMF global variables.

### **Related reference**

[Default views that are used for the commands](#)

QMF provides default views during installation. How each view is used depends on whether you have issued the LIST or DESCRIBE command as well as the location to which the command is directed.

## Default behavior of the QMF LIST and DESCRIBE commands

When a user issues a LIST TABLES command or a DESCRIBE command for a column in a table, QMF generates the required information from views that are defined on a set of database catalog tables. QMF global variables are used to store the view names.

### Default views that are used for the commands

QMF provides default views during installation. How each view is used depends on whether you have issued the LIST or DESCRIBE command as well as the location to which the command is directed.

The SQL statement that creates the view is shown beneath each view name. These view definitions can also be found in job DSQ0BCTV, which is run when you run job DSQ1BVW during installation.

The default views can return multiple identical rows if SYSIBM.SYSTABAUTH has multiple entries authorizing the user or PUBLIC to a given table. When used by the QMF LIST or DESCRIBE commands, rows with duplicate OWNER and TNAME (for the table view) or duplicate OWNER, TNAME, and CNAME (for the column view) are ignored.

### Q.DSQEC\_TABS\_LDB2L

**Note:** Depending on the value of installation parameter EXTSEC, the definitions of views Q.DSQEC\_TABS\_LDB2L, Q.DSQEC\_COLS\_LDB2L, Q.DSQEC\_TABS\_RDB2L and Q.DSQEC\_COLS\_RDB2L created at installation time may not include reference to the SYSIBM.SYSTABAUTH. EXTSEC installation parameter, which is set in the QMF1210.SDSQEXCE(DSQ1DEFS) exec and is referenced by the QMF1210.SDSQSAPE(DSQ1BVW) job. EXTSEC indicates whether internal or externity security control is in place.

The view Q.DSQEC\_TABS\_LDB2L is used for LIST commands directed to the current location (the database to which you are currently connected). This view is used for all database types except DB2 for VSE and VM.

The following SQL statements create the view. These statements apply to Db2 for z/OS databases only. The DDL for Db2 for Linux, UNIX, and Windows and DB2 for iSeries can be found in job DSQ0BCTV.

```
CREATE VIEW Q.DSQEC_TABS_LDB2L
  (OWNER, TNAME, TYPE, SUBTYPE, MODEL, RESTRICTED, REMARKS,
   CREATED, MODIFIED, LAST_USED, LABEL, LOCATION, OWNER_AT_LOCATION,
   NAME_AT_LOCATION)
AS SELECT
  VARCHAR(RTRIM(T.CREATOR)),
  T.NAME, 'TABLE', T.TYPE
  , , , ,
  VARCHAR(RTRIM(SUBSTR(T.REMARKS, 1, 254))),
  VARCHAR(RTRIM(SUBSTR(T.LABEL, 1, 30))),
  SUBSTR(T.LOCATION, 1, 16),
  VARCHAR(RTRIM(T.TBCREATOR)), T.TBNAME
FROM SYSIBM.SYSTABLES T
, ( SELECT DISTINCT
  VARCHAR(RTRIM(TA.TCREATOR)), TA.TTNAME
  FROM SYSIBM.SYSTABAUTH TA
  WHERE TA.GRANTEETYPE=' '
  AND TA.GRANTEE IN (
  USER, CURRENT SQLID, CURRENT SCHEMA,
  'PUBLIC', 'PUBLIC*')
  ) AS UAT ("CREATOR", "NAME")
WHERE (T.CREATOR=UAT.CREATOR OR
  T.CREATOR=CURRENT SCHEMA)
  AND T.NAME=UAT.NAME
  AND T.TYPE IN ('T', 'V', 'H');
```

### Q.DSQEC\_TABS\_RDB2L

The view Q.DSQEC\_TABS\_RDB2L is used for LIST commands that include the LOCATION parameter. LIST commands that include the LOCATION parameter can be initiated from and directed to Db2 for z/OS databases only.

The following SQL statements create the view:

```
CREATE VIEW Q.DSQC_TABS_RDB2L
(OWNER, TNAME, TYPE, SUBTYPE, MODEL,
RESTRICTED, REMARKS, CREATED, MODIFIED,
LAST_USED, LABEL, LOCATION, OWNER_AT_LOCATION,
NAME_AT_LOCATION)
AS SELECT VARCHAR(RTRIM(T.CREATOR)), T.NAME, 'TABLE', T.TYPE
        VARCHAR(RTRIM(SUBSTR(T.REMARKS, 1, 254))),
        VARCHAR(RTRIM(SUBSTR(T.LABEL, 1, 30))),
        SUBSTR(T.LOCATION, 1, 16),
        VARCHAR(RTRIM(T.TBCREATOR)), T.TBNAME
FROM SYSIBM.SYSTABLES T
, ( SELECT DISTINCT
      VARCHAR(RTRIM(TA.TCREATOR)), TA.TTNAME
    FROM SYSIBM.SYSTABAUTH TA
    WHERE TA.GRANTEETYPE=' '
    AND TA.GRANTEE IN (
      USER, CURRENT SQLID, CURRENT SCHEMA, 'PUBLIC*' )
    ) AS UAT ("CREATOR", "NAME")
WHERE (T.CREATOR=UAT.CREATOR OR
       T.CREATOR=CURRENT SCHEMA)
       AND T.NAME=UAT.NAME
       AND T.TYPE IN ('T', 'V', 'H');
```

### Q.DSQC\_COLS\_LDB2L

The view Q.DSQC\_COLS\_LDB2L is used for DESCRIBE commands issued for tables at the current location (the database to which you are currently connected). This view is used for all database types except DB2 for VSE and VM.

The following SQL statements create the view. These statements apply to Db2 for z/OS databases only. The DDL for Db2 for Linux, UNIX, and Windows and DB2 for iSeries can be found in job DSQ0BCTV.

```
CREATE VIEW Q.DSQC_COLS_LDB2L
( OWNER, TNAME, CNAME, REMARKS, LABEL )
AS SELECT
      VARCHAR(RTRIM(C.TBCREATOR)),
      C.TBNAME,
      VARCHAR(RTRIM(SUBSTR(C.NAME, 1, 30))),
      VARCHAR(RTRIM(SUBSTR(C.REMARKS, 1, 254))),
      VARCHAR(RTRIM(SUBSTR(C.LABEL, 1, 30)))
FROM SYSIBM.SYSCOLUMNS C
, ( SELECT DISTINCT
      VARCHAR(RTRIM(TA.TCREATOR)), TA.TTNAME
    FROM SYSIBM.SYSTABAUTH TA
    WHERE TA.GRANTEETYPE=' '
    AND TA.GRANTEE IN (
      USER, CURRENT SQLID, CURRENT SCHEMA,
      'PUBLIC', 'PUBLIC*' )
    ) AS UAT ("CREATOR", "NAME")
WHERE (C.TBCREATOR=UAT.CREATOR OR
       C.TBCREATOR=CURRENT SCHEMA)
       AND C.TBNAME=UAT.NAME;
```

### Q.DSQC\_COLS\_RDB2L

The view Q.DSQC\_COLS\_RDB2L is used for DESCRIBE commands for Db2 for z/OS databases only.

The following SQL statements create the view:

```
CREATE VIEW Q.DSQC_COLS_RDB2L
( OWNER, TNAME, CNAME, REMARKS, LABEL )
AS SELECT VARCHAR(RTRIM(C.TBCREATOR)),
        C.TBNAME,
        VARCHAR(RTRIM(SUBSTR(C.NAME, 1, 30))),
        VARCHAR(RTRIM(SUBSTR(C.REMARKS, 1, 254))),
        VARCHAR(RTRIM(SUBSTR(C.LABEL, 1, 30)))
FROM SYSIBM.SYSCOLUMNS C
, ( SELECT
      VARCHAR(RTRIM(TA.TCREATOR)), TA.TTNAME
    FROM SYSIBM.SYSTABAUTH TA
    WHERE TA.GRANTEETYPE=' '
    AND TA.GRANTEE IN (
```

```

        USER,CURRENT SQLID,'PUBLIC*'
    )
  ) AS UAT ("CREATOR","NAME")
  WHERE C.TBCREATOR=UAT.CREATOR AND C.TBNAME=UAT.NAME

```

### Q.DSQEC\_ALIASESL

The view Q.DSQEC\_ALIASESL selects only the list of aliases for a list of tables, or the column information for an alias, for all database types except DB2 for VSE and VM.

The following SQL statements create the view. These statements apply to Db2 for z/OS databases only. The DDL for Db2 for Linux, UNIX, and Windows and DB2 for iSeries can be found in job DSQ0BCTV.

```

CREATE VIEW Q.DSQEC_ALIASESL
( OWNER, TNAME, TYPE, SUBTYPE, MODEL,
  RESTRICTED, REMARKS, CREATED, MODIFIED,
  LAST_USED, LABEL, LOCATION,
  OWNER_AT_LOCATION, NAME_AT_LOCATION )
AS SELECT
  VARCHAR(RTRIM(T.CREATOR)),T.NAME,'TABLE',T.TYPE
  ,' ',' ',' ',VARCHAR(RTRIM(SUBSTR(T.REMARKS,1,254))),
  VARCHAR(RTRIM(SUBSTR(T.LABEL,1,30))),
  SUBSTR(T.LOCATION,1,16),
  VARCHAR(RTRIM(T.TBCREATOR)),T.TBNAME
FROM SYSIBM.SYSTABLES T
  WHERE T.CREATOR
  IN (USER,CURRENT SQLID,CURRENT SCHEMA)
  AND T.TYPE = 'A';

```

### Q.DSQEC\_TABS\_SQLL

The view Q.DSQEC\_TABS\_SQLL is used for LIST commands that are directed to a DB2 for VSE and VM database to which you are currently connected.

The following SQL statement creates the view:

```

CREATE VIEW Q.DSQEC_TABS_SQLL
(OWNER,TNAME,TYPE,SUBTYPE,MODEL,RESTRICTED,REMARKS,
  CREATED,MODIFIED,LAST_USED,LABEL,LOCATION,
  OWNER_AT_LOCATION,NAME_AT_LOCATION)
AS SELECT
  STRIP(CREATOR),TNAME,'TABLE',TABLETYPE,' ',' ',' ',
  REMARKS,' ',' ',' ',' ',' ',
  TLABEL,' ',' ',' ',' ',' '
FROM SYSTEM.SYSCATALOG,SYSTEM.SYSTABAUTH
  WHERE CREATOR = TCREATOR AND TNAME=TTNAME AND
  GRANTEETYPE = ' ' AND
  GRANTEE IN (USER,'PUBLIC')

```

### Q.DSQEC\_COLS\_SQLL

The view Q.DSQEC\_COLS\_SQLL is used for DESCRIBE commands that are directed to a DB2 for VSE and VM database to which you are currently connected.

The following SQL statement creates the view:

```

CREATE VIEW Q.DSQEC_COLS_SQLL
(OWNER,TNAME,CNAME,REMARKS,LABEL)
AS SELECT
  STRIP(CREATOR),TNAME,CNAME,REMARKS,CLABEL
FROM SYSTEM.SYSCOLUMNS,SYSTEM.SYSTABAUTH
  WHERE TCREATOR = CREATOR AND TTNAME = TNAME AND
  GRANTEETYPE = ' ' AND GRANTEE IN (USER,'PUBLIC')

```

### Global variables that store the default view names

The names of the default views used in LIST and DESCRIBE commands are stored in QMF global variables.

#### For LIST commands:

- If the command is directed to the local Db2 for z/OS database (or if you issue a CONNECT command to connect to a remote database other than DB2 for VSE and VM and then issue a LIST TABLES



command), QMF uses the views named in the DSQEC\_ALIASES and DSQEC\_TABS\_LDB2 global variables. By default, the DSQEC\_ALIASES global variable is set to Q.DSQEC\_ALIASESL and the DSQEC\_TABS\_LDB2 global variable is set to Q.DSQEC\_TABS\_LDB2L.

- If you issue a LIST TABLES command that includes the LOCATION parameter, QMF uses the view named in the DSQEC\_TABS\_RDB2 global variable, which defaults to the view Q.DSQEC\_TABS\_RDB2L. LIST commands with the LOCATION parameter can be initiated from and directed to Db2 for z/OS databases only.
- If you issue a CONNECT command to connect to a DB2 for VSE and VM database and then issue a LIST TABLES command, QMF uses the view named in the DSQEC\_TABS\_SQL global variable, which is Q.DSQEC\_TABS\_SQLL by default.

#### **For DESCRIBE commands:**

When you issue a DESCRIBE command, the views that QMF uses to generate the required information also depend on the database to which the command is directed:

- If the command is directed to the local Db2 for z/OS database (or if you issue a CONNECT command to connect to a remote database other than DB2 for VSE and VM and then issue the DESCRIBE command), QMF uses the view named in the DSQEC\_COLS\_LDB2 global variable. This variable is set to Q.DSQEC\_COLS\_LDB2L by default.
- If you issue a LIST TABLES command that includes the LOCATION parameter and then issue the DESCRIBE command for one of the tables or views in the list, QMF uses the view named in the DSQEC\_COLS\_RDB2 global variable. This variable is set to Q.DSQEC\_COLS\_RDB2L by default. LIST commands with the LOCATION parameter can be initiated from and directed to Db2 for z/OS databases only.
- If you issue a CONNECT command to connect to a DB2 for VSE and VM database and then issue a LIST TABLES command followed by a DESCRIBE command for one of the tables or views in the list, QMF uses the views named in the DSQEC\_COLS\_SQL global variable. This variable is set to Q.DSQEC\_COLS\_SQLL by default.

#### **Related reference**

Default views that are used for the commands

QMF provides default views during installation. How each view is used depends on whether you have issued the LIST or DESCRIBE command as well as the location to which the command is directed.

## **Object list storage requirements**

For the LIST command, there are two sets of storage requirements for each row of the object list.

These are the storage requirements for each row:

- The QMF internal RPT record collection requires:
  - Object OWNER key information (50 bytes)
  - REMARKS (up to 254 bytes)
  - TABLE with a LABEL (up to 30 bytes)
  - ALIAS (42 bytes)
  - Object information for QUERY, PROC, FORM, and ANALYTIC (63 bytes)
- Displayed data and control information requires 130 bytes plus the actual number of bytes for REMARKS (up to 254 bytes) and the actual number of bytes for the LABEL associated with a table (up to 30 bytes).

#### **Related reference**

QMF views

These views are shipped with QMF. QMF uses these views on the platforms indicated in the table to create object lists when the QMF LIST command is issued.



---

# Chapter 12. Creating and maintaining objects in the database

The way in which you set up and maintain QMF and database objects can impact how easily your users are able to use QMF as well as the security of your data and the performance of your system.

---

## Enabling users to create tables in the database

Depending on the needs of your site, you might need to create tables for your users or enable them to create their own tables.

A QMF user can create a table using any of these methods:

- SQL CREATE TABLE statement

Run the SQL CREATE TABLE statement from the **SQL Query** panel or directly from the database.

**Tip:** A single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command.

- QMF DISPLAY command, followed by the SAVE DATA command

Information about the syntax and options of the SAVE DATA command is included in the description of the SAVE command.

- QMF IMPORT command

### Procedure for creating tables

The steps for creating a table varies depending on whether you are creating the tables for your users or users are creating their own tables.

#### Procedure

- If you are creating tables for your users, follow these steps:
  - a) Create a table space and define it to the database before its first use.
  - b) To create the table, issue either an SQL CREATE TABLE statement, a QMF DISPLAY command followed by a SAVE DATA command, or an IMPORT command.
  - c) To improve Db2 performance, create one or more indexes on the tables you create.
  - d) Fill the tables with data.

Use the Db2 for z/OS LOAD utility, QMF IMPORT commands (for transferring small tables), or other methods.
  - e) Grant Db2 and SQL privileges for the tables to users who need them.
- If users are creating tables themselves, follow these steps:
  - a) Grant a user Db2 CREATETS or CREATETAB authority. If you choose to grant only CREATETAB authority, create a table space and define it to the database before its first use.
  - b) Assign the table space in the user's QMF profile, using an SQL UPDATE statement for the SPACE field.

You can update the SYSTEM profile if you need to change its default values.
  - c) Grant CREATETAB authority to users creating their own tables in table spaces, or assign CREATETS authority and allow users to create table spaces for their own use.

Users automatically have all SQL privileges on tables and table spaces they create.

- d) Provide education on the SQL CREATE TABLE statement, QMF SAVE DATA and IMPORT commands, and other guidelines your site has for creating tables.
- e) Grant Db2 and SQL privileges on any table or view on which users issue SAVE DATA or IMPORT commands to create new tables.

Grant at least the SELECT privilege, or QMF cannot read the data to create a new table.

## Assigning a table space for SAVE DATA, IMPORT, and RUN QUERY commands

Where tables are saved in the database when users issue SAVE DATA, IMPORT TABLE or RUN QUERY commands depends on the setting of the SPACE option of the QMF™ profile or the value of the SPACE keyword on the command.

### Valid values for the SPACE field of the profile

Users using the QMF SYSTEM profile have a default database and table space of DSQDBDEF.DSQTSDEF. Newer installations will have a default SPACE value of DATABASE "DSQDBDEF". This value may be overridden by using the keyword value SPACE on the commands.

You can set the SPACE option of the profile with the following values:

- *databasename.spacename*

Specify both a database and a table space to have QMF save the table in that specific table space. Throughout this topic, this is known as an explicit table space. The table space that you specify must have already been created. QMF will not create the table space.

For example, the following query configures the profile of a QMF user named SMITH to save tables in table space TSPACE1, in database DBASE1:

```
UPDATE Q.PROFILES
SET SPACE='DBASE1.TSPACE1'
WHERE CREATOR='SMITH' AND TRANSLATION='ENGLISH'
```

However, if all users are using the same table space, you could experience resource contention.

- DATABASE "*databasename*"

When you specify the DATABASE keyword followed by a database name in quotes, each table is created in a separate table space created implicitly and exclusively for that table by Db2. Such implicit table spaces have default LOCKSIZE, BUFFERPOOL, STOGROUP, and space attributes, and have names that match the name of the table that was created. QMF recommends this method.

For example, the following value for the SPACE field of the Q.PROFILES table saves each table in a separate table space in the DSQDBDEF database:

```
DATABASE "DSQDBDEF"
```

Limiting each table space to one table only minimizes resource contention. Both partitioned and universal table space schemes impose a limit of one table per table space. QMF recommends this method for SAVE DATA operations.

- NULL or blank

If the SPACE field of the QMF profile is null or blank, SAVE DATA or IMPORT TABLE commands create tables in implicitly generated table space names in the DSNDB04 database by default.

### Factors to consider when choosing a table space scheme

Consider these factors when you decide whether to explicitly create table spaces or configure QMF to implicitly create them when they are needed:

#### Table sizes

The default attributes for implicitly created table spaces might not be suitable for the intended tables. The default values for the space parameters (PRIQTY and SECQTY) are intended for small sample and

summary tables. If the user's tables are large, the explicit table space option is probably the better choice.

If the table space is too small, the new table remains in the table space but is empty. The table space must therefore be enlarged before the SAVE or IMPORT commands can run successfully.

### **Maintenance**

When you use the QMF explicit table space option, you simplify maintenance if you take advantage of segmented table spaces. When a table is dropped in a segmented table space, its segments become immediately available for reuse when the drop is committed. It is not necessary to wait for reorganization of the table space.

Implicitly created table spaces can also simplify maintenance. An implicitly created table space is erased automatically when the table it contains is erased.

QMF for TSO and CICS recommends using implicitly created universal table spaces for SAVE DATA, IMPORT TABLE, and RUN QUERY operations. For new QMF installations, this is the default method of operation.

Choose the method which best conforms to your site's standards.

### **Resource contention**

To avoid resource contention, use either the explicit table space option with a segmented table space or implicitly created, universal tablespaces. With a segmented table space, when a table is locked, the lock does not interfere with access to segments of other tables.

### **Integrity and security**

You might have to grant the user certain Db2 privileges that the user would not otherwise need. With the explicit table space option, you can limit these added privileges to the creation of tables in the chosen table space. With the implicit table space option, you must grant the user the privilege to create table spaces for the database, and you cannot restrict this privilege to table spaces created with the SAVE and IMPORT commands.

### **Convenience**

An explicitly created table space named DSQDBDEF.DSQTSDEF is the default table space for users using the QMF SYSTEM profile. It is created during QMF installation and is used in the installation verification procedures. You might find that this table space is large enough to hold the tables of your users.

This table space should be used by multiple users only if the tables are primarily read-only; otherwise resource contention could prevent necessary updates.

## **Granting a user the privileges to create tables**

The authority that you must grant to a user who creates tables varies. One set of privileges are required when you specify a database and table space name in the SPACE field of the QMF user profile. A different set of privileges is required when you specify only a database or no value.

### **Privileges that are required for explicitly created table spaces:**

At a minimum, the user needs the CREATETAB privilege for the database and the USE privilege on the receiving table space.

If you want to allow a user to create tables, but want to maintain control over how much resource is used, assign a table space for the user rather than granting CREATETS authority. That way, you can control the size of the table space and the amount of resource used.

### **Privileges that are required for implicitly created table spaces:**

At a minimum, the user needs the CREATETAB and CREATETS privileges on the database.

Users of the default Db2 for z/OS database, DSNDB04, might already have some of the preceding privileges. During database installation, the CREATETAB and CREATETS privileges for the default database are granted to PUBLIC. A user of the default database, operating under the implicit table space option, automatically has the minimum authority to create tables. If, instead, this user operates under the explicit table space option, only the USE privilege must be granted.

**Important:** Do not grant privileges to any of the databases that are used exclusively by Db2 itself. These databases include DSNDB01, DSNDB03, and DSNDB05.

When users create tables for others, the owner qualifier (the owner of the object) must be the user's primary or secondary authorization ID. With other IDs, the appropriate CREATE table statement might run, but INSERT statements might not run. When users create their own tables after the table structure is created, the users have the necessary INSERT privilege. All that is needed is the privilege to run the CREATE TABLE statement.

### Related concepts

[Valid values for the SPACE field of the profile](#)

## Using views to filter sensitive data

---

You can use views to filter sensitive data from certain users.

### Creating a view

Use the CREATE VIEW statement to create a view.

For example, suppose that you want to create a view that is based on the table SMITH.STAFF, which contains personnel information. Each row in the table represents an employee. For each row, you want the view to show the employee's name, department, job classification, and years of service. You do not want it to show the employee's salary and commission.

You can create the view with a SQL statement like the following one.

```
CREATE VIEW VIEWA AS
  SELECT NAME, DEPT, JOB, YEARS
  FROM SMITH.STAFF
```

To create a view, the user's SQL authorization ID must have, at a minimum, the SELECT privilege on each of the underlying objects of the view.

If the owner of a view loses the SELECT privilege on one or more of the underlying objects, the view is dropped from the system. Any views that use that dropped view as an underlying object are also dropped, and so on.

### Granting privileges for a view

As with tables, you can grant SELECT, INSERT, UPDATE, and DELETE privileges on a view.

With the SELECT privilege, a person can use the view just like a table in SELECT queries and subqueries. With the other privileges, a person can modify the data in the table underlying the view.

Granting a privilege for a view begins with the owner of the view. The privileges the owner can grant depend on the privileges the owner has on the underlying objects of the view. These are the tables and views that are named in the FROM clause of the view's defining CREATE statement. For example, the underlying object of the view created with this statement is the table SMITH.STAFF:

```
CREATE VIEW VIEWA AS
  SELECT NAME, DEPT, JOB, YEARS
  FROM SMITH.STAFF
```

When the underlying objects include views, or objects not owned by the owner of the view, the privileges the owner holds on the underlying objects might vary widely. In this situation, the following rules apply:

- The owner of a view always has the SELECT privilege on the view. The owner has this privilege with the GRANT option if the owner has the SELECT privilege with the GRANT option on each of the underlying objects of the view.
- The owner of a view has the INSERT, UPDATE, or DELETE privileges on the view if both of the following are true:
  - The view is not read-only. This implies that the view has a single underlying object.

A view is automatically read-only if the statement that created it joins two or more tables.

- The owner of the view has the same privilege on the underlying object.

As with tables, the WITH GRANT OPTION clause can be specified so that the grantee can grant privileges to others.

## Maintaining the QMF object catalog

Periodically, you must condense and reorganize the QMF object catalog, which stores information about QMF queries, forms, procedures, analytics, and folders.

Regular maintenance of the QMF catalog might involve tasks such as transferring objects to new owners or enlarging the table space for the tables when it is no longer large enough to hold existing QMF objects.

All QMF catalog information about queries, forms, procedures, analytics, and folder objects is stored among three QMF control tables:

- Q.OBJECT\_DIRECTORY
- Q.OBJECT\_DATA
- Q.OBJECT\_REMARKS

You can keep QMF and the database running efficiently by periodically listing, displaying, or deleting QMF objects from these tables and reorganizing them when necessary. You might also need to use the information in these tables to transfer an object from one owner to another. You must assign STATS and REORG privileges to a user who is monitoring or reorganizing the QMF catalog control tables.

### Related reference

[QMF control tables and table spaces for TSO and CICS](#)  
These are the control tables shipped with QMF.

## Structure of the Q.OBJECT\_DIRECTORY table

The Q.OBJECT\_DIRECTORY table contains a row for each QMF query, form, procedure, folder, and analytics objects in the database.

The table has the index Q.OBJECT\_DIRECTORYX, with the UNIQUE and CLUSTER attributes. The keyed columns are OWNER and NAME.

The Q.OBJECT\_DIRECTORY table has the structure shown in the following table:

| Column name | Data type and length                                                                                                                 | Nulls allowed? | Function/values                                          |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------------------|
| OWNER       | VARCHAR(128)<br><b>Exception:</b> The OWNER column is defined as CHAR(8) when you are connected to a DB2 for VSE and VM database.    | No             | Shows the authorization ID of the creator of the object. |
| NAME        | VARCHAR(128)<br><b>Exception:</b> The NAME column is defined as VARCHAR(18) when you are connected to a DB2 for VSE and VM database. | No             | Shows the name of the object.                            |

Table 44. Structure of the Q.OBJECT\_DIRECTORY table (continued)

| Column name | Data type and length | Nulls allowed? | Function/values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------|----------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPE        | CHAR(8)              | No             | Shows the type of object: FORM, PROC, ANALYTIC, QUERY, or FOLDER.<br><b>Exception:</b> DB2 Server for VSE and VM.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| SUBTYPE     | CHAR(8)              | Yes            | The following subtypes are used for QUERY objects: <ul style="list-style-type: none"> <li>• SQL</li> <li>• QBE</li> <li>• PROMPTED</li> </ul> The following subtypes are used for ANALYTIC objects: <p><b>PLOT</b><br/>Plot chart</p> <p><b>HIST</b><br/>Histogram</p> <p><b>PIE</b><br/>Pie chart</p> <p><b>TOWER</b><br/>Tower chart</p> <p><b>MIXED</b><br/>Mixed chart</p> <p><b>MAP</b><br/>Map</p> <p><b>UNIVAR</b><br/>Univariate curve</p> <p><b>LINEAR</b><br/>Linear trend</p> <p><b>DCF</b><br/>Discounted cash flow</p> <p><b>BASIC</b><br/>Basic statistics</p> <p><b>BIVAR</b><br/>Bivariate curve</p> <p><b>WILCOSR</b><br/>Wilcoxon Signed-Rank Test</p> <p><b>MANNW</b><br/>Mann-Whitney U Test</p> This field is blank for all other object types. |
| OBJECTLEVEL | INTEGER(4)           | No             | QMF uses this number to reconstruct an object from its defining text in the Q.OBJECT_DATA table.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |



Table 44. Structure of the Q.OBJECT\_DIRECTORY table (continued)

| Column name | Data type and length | Nulls allowed? | Function/values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|----------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RESTRICTED  | CHAR(1)              | No             | YES if the object has not been shared (using the SHARE parameter of the QMF SAVE command or the SHARE option of the SAVE panel in QMF Analytics for TSO); NO if the object has been shared with other users.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| MODEL       | CHAR(8)              | Yes            | This value is always REL, indicating relational data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| CREATED     | TIMESTAMP            | Yes            | Shows the timestamp value for when an object was created. The value is recorded after SAVE or IMPORT commands.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| MODIFIED    | TIMESTAMP            | Yes            | Shows the timestamp value for when an object was last modified. The value is recorded after SAVE or IMPORT commands.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| LAST_USED   | TIMESTAMP            | Yes            | <p>Shows the date value for when an object was last used, regardless of whether the command that accessed the object was successful. However, in some cases, the LAST_USED column is not updated immediately after a command is issued, and if QMF is terminated abnormally, the column might not be updated.</p> <p>By default, the following commands cause the LAST_USED value to change:</p> <ul style="list-style-type: none"> <li>• CONVERT</li> <li>• DISPLAY</li> <li>• EXPORT</li> <li>• IMPORT</li> <li>• LAYOUT</li> <li>• PRINT</li> <li>• RUN</li> <li>• SAVE</li> </ul> <p>You can restrict the LAST_USED column to updates by RUN, SAVE, and IMPORT commands only by setting the DSQEC_LAST_RUN global variable to 1.</p> <p>You can also restrict the LAST_USED column to updates by the RUN command only by setting the DSQEC_LAST_RUN global variable to 2.</p> <p>The value is updated only once a day, even if the object is used multiple times.</p> |

## Structure of the Q.OBJECT\_DATA table

The Q.OBJECT\_DATA table contains one or more rows for each query, form, procedure, folder, and analytics object in the database.

Each row contains all or part of the defining text for each object. Objects are reconstructed from this text by combining the text with the corresponding format number in the OBJECTLEVEL column of the Q.OBJECT\_DIRECTORY table.

The Q.OBJECT\_DATA table has the index Q.OBJECT\_OBJDATA, with the UNIQUE and CLUSTER attributes. Keyed columns are OWNER, NAME, and SEQ.

The table has the structure shown in the following table:

*Table 45. Structure of the Q.OBJECT\_DATA table*

| Column name | Data type and length                                                                                                                 | Nulls allowed? | Function/values                                                                                                                                                                                                                               |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OWNER       | VARCHAR(128)<br><b>Exception:</b> The OWNER column is defined as CHAR(8) when you are connected to a DB2 for VSE and VM database.    | No             | Shows the authorization ID of the creator of the object.                                                                                                                                                                                      |
| NAME        | VARCHAR(128)<br><b>Exception:</b> The NAME column is defined as VARCHAR(18) when you are connected to a DB2 for VSE and VM database. | No             | Shows the name of the object.                                                                                                                                                                                                                 |
| TYPE        | CHAR(8)                                                                                                                              | No             | Shows the type of object: FORM, PROC, ANALYTIC, QUERY, or FOLDER.<br><b>Exception:</b> FOLDER is not a valid TYPE when QMF is connected to DB2 Server for VSE and VM.                                                                         |
| SEQ         | SMALLINT(2)                                                                                                                          | No             | Indicates the sequence that this text occupies within the entire text of the object. For example, if this row is the first row of text in the object, SEQ is 1; if it is the second, SEQ is 2, and so on.                                     |
| APPLDATA    | VARCHAR(3600)<br>FOR BIT DATA                                                                                                        | Yes            | Contains the data that constitutes the object.<br><b>Important:</b> Because the data is stored in binary format, do not try to update this column. Additionally, the APPLDATA column must never be subjected to code page (CCSID) conversion. |

## Structure of the Q.OBJECT\_REMARKS table

The Q.OBJECT\_REMARKS table contains one row for each query, form, procedure, folder, and analytics object in the database.

Each row contains comments that are entered when the QMF SAVE command is issued or when the Save function key is used in QMF Analytics for TSO.

The Q.OBJECT\_REMARKS table has the index Q.OBJECT\_REMARKSX, with the UNIQUE and CLUSTER attributes. Keyed columns are OWNER and NAME.

The table has the structure shown in the following table:

*Table 46. Structure of the Q.OBJECT\_REMARKS table*

| Column name | Data type and length                                                                                                                 | Nulls allowed? | Function/values                                                                                                                                                           |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OWNER       | VARCHAR(128)<br><b>Exception:</b> The OWNER column is defined as CHAR(8) when you are connected to a DB2 for VSE and VM database.    | No             | Shows the authorization ID of the user who created the object.                                                                                                            |
| NAME        | VARCHAR(128)<br><b>Exception:</b> The NAME column is defined as VARCHAR(18) when you are connected to a DB2 for VSE and VM database. | No             | Shows the name of the object.                                                                                                                                             |
| TYPE        | CHAR(8)                                                                                                                              | No             | Shows the type of the object: FORM, PROC, ANALYTIC, QUERY, or FOLDER.<br><b>Exception:</b> FOLDER is not a valid TYPE when QMF is connected to DB2 Server for VSE and VM. |
| REMARKS     | VARCHAR(254)                                                                                                                         | Yes            | Contains the comment that was saved with the object when it was created or replaced.                                                                                      |

## Enlarging the table space for the QMF object catalog

Periodically, QMF objects might collectively become too large for the table spaces that contain the QMF object catalog control tables (Q.OBJECT\_DIRECTORY, Q.OBJECT\_DATA, and Q.OBJECT\_REMARKS).

### Before you begin

To help you estimate the amount of space needed, you can determine the amount of space currently used. If the space is managed by Db2, you can get this information in the following way:

1. Run the STOSPACE utility on the table space's storage group.
2. Run the following query:

```
SELECT SPACE
FROM SYSIBM.SYSTABLEPART
WHERE TSNAME='ttttttt' AND DBNAME='DSQDBCTL'
```

In this statement, *ttttttt* is the table space name. The result (SPACE) gives the number of kilobytes of storage currently allocated to the table space.

### About this task

The default table spaces are listed in the following table.

In addition to this procedure, you can also use the Db2 LOAD utility to enlarge a table space.

*Table 47. Table spaces for control tables that store information about QMF objects*

| Table space name | Contents                                | Default size |
|------------------|-----------------------------------------|--------------|
| DSQTSCT1         | Q.OBJECT_DIRECTORY table                | 256 pages    |
| DSQTSCT2         | Q.OBJECT_REMARKS table                  | 256 pages    |
| DSQTSCT3         | Q.OBJECT_DATA and Q.OBJECT_DATA2 tables | 5120 pages   |

**Important:** QMF Version 12.1 creates table space data sets managed by Db2 if QMF was not previously installed. Do not change the QMF storage groups from managed by Db2 to user-managed after QMF installation. However, if the space is user-managed, you can use the TSO LISTCAT command for the space information if you know the data set name.

### Procedure

To enlarge the table space for the QMF object catalog control tables, follow these steps:

1. Make an image copy of the table space.

You can use this for restoration if the procedure fails.

2. Create a storage group for the table space.

Do this only if the table space has user-managed data sets, and no storage group is already available.

To determine the type of data set management used for the table space, run the following query:

```
SELECT STORTYPE
FROM SYSIBM.SYSTABLEPART
WHERE TSNAME='DSQTSCT3' AND DBNAME='DSQDBCTL'
```

This query produces a one-line result for the table space DSQTSCT3. In the result, STORTYPE has the value E or I:

**E**

Indicates that the data sets for the table space are user-managed (no associated storage group).

**I**

Indicates that the data sets for the table space are managed by Db2. The following table shows the default database partition groups for the QMF control tables.

*Table 48. Database partition groups for control tables that store information about QMF objects*

| Database Partition Group Name | Used for                                                                | Characteristics                                                                  |
|-------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| DSQTSCTL                      | For all QMF control tables except as described elsewhere in this table. | Can be distributed across multiple database partitions. Growth potential is low. |

| Table 48. Database partition groups for control tables that store information about QMF objects (continued) |                                                                                                               |                                                                                           |
|-------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Database Partition Group Name                                                                               | Used for                                                                                                      | Characteristics                                                                           |
| DSQTSOBJ                                                                                                    | The QMF object catalog control tables where procedure, query, form, folder, and analytics objects are stored. | Can be distributed across multiple database partitions. Growth potential is high.         |
| DSQTSDEF                                                                                                    | The default SAVE DATA space as initialized in the QMF profile.                                                | Should be defined to be restricted to a single database partition to avoid complications. |
| DSQTSAMP                                                                                                    | The QMF sample tables.                                                                                        | Can be distributed across multiple database partitions.                                   |

3. Stop the database using the following command:

```
-STOP DATABASE(DSQDBCTL)
```

4. Change the table space description.

- If the table space data sets are user-managed, issue a Db2 statement like the following example:

```
ALTER TABLESPACE DSQDBCTL.tttttt
  USING STOGROUP ssssss PRIQTY pppp SECQTY ssss
```

In this statement, *tttttt* is the table space name. The statement changes the table space from user-managed to managed by Db2 and names a storage group (*ssssss*) for the management. The quantities *pppp* and *ssss* are the new primary and secondary allocation sizes (in kilobytes) for the enlarged table space.

- If the table space data sets are managed by Db2, execute a Db2 statement like the following:

```
ALTER TABLESPACE DSQDBCTL.tttttt
  PRIQTY pppp SECQTY ssss
```

In this statement, *tttttt* is the table space name. The character strings *pppp* and *ssss* represent the new primary and secondary allocation sizes, in kilobytes, for the enlarged table space.

5. Move the table space data.

Simply changing the table space description does not affect enlargement. You must instead do something that causes the table space to be refilled.

6. Start the database with the statement:

```
-START DATABASE(DSQDBCTL)
```

## Listing QMF objects

To get the information you need to help you maintain the QMF environment, you can list the queries, forms, procedures, folders, and analytic objects that QMF users have saved in the database.

With administrator authority, you can list QMF objects that you do not own by using the following query.

```

SELECT D.NAME, D.TYPE, D.SUBTYPE, D.RESTRICTED, R.REMARKS
FROM Q.OBJECT_DIRECTORY D,
Q.OBJECT_REMARKS R
WHERE D.OWNER = 'userid'
AND D.OWNER = R.OWNER
AND D.NAME = R.NAME
ORDER BY D.TYPE, D.SUBTYPE, D.RESTRICTED

```

Figure 34. Listing queries, forms, procedures, folders and analytic objects that are owned by a particular user

This query returns a list of objects that is sorted by type (ANALYTICS, FOLDER, FORM, PROC, or QUERY) and further by subtype if type is ANALYTICS or QUERY. Objects of each type are further sorted by whether they have been shared by the owner. Shared status is reflected in the RESTRICTED column of the Q.OBJECT\_DIRECTORY table; a value of "Y" indicates that the object is not shared.

## Displaying QMF objects

In addition to listing a query, procedure, form, or analytic object to get the information you need to help you maintain the QMF environment, you can use the DISPLAY command for more information.

DISPLAY is not a valid command for QMF folder objects because folders do not contain object data. Folders contain only references to other QMF objects (queries, procedures, forms, or analytic objects).

If listing the objects does not provide enough information in the REMARKS column, try displaying the object using one of the following methods:

- Issue the following statement to share the user's objects. Then display them from your own ID:

```

UPDATE Q.OBJECT_DIRECTORY
SET RESTRICTED = 'N'
WHERE OWNER = 'userid'

```

**Important:** Issue this statement only if you do not need to track which of the user's objects are restricted and which are not. After you issue this statement, you can reset RESTRICTED to Y, but then you will not be able to tell which objects were originally restricted.

**Tip:** A single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command.

- Issue the QMF DISPLAY command for each object that you want to display.

### Related concepts

#### Listing QMF objects

To get the information you need to help you maintain the QMF environment, you can list the queries, forms, procedures, folders, and analytic objects that QMF users have saved in the database.

## Transferring ownership of QMF objects

You must run three statements to transfer QMF objects (queries, forms, procedures, folders, and analytic objects) from one user to another. Ensure that you run all three statements.

**Important:** First ensure that the new owner has no objects that are saved with the name of the object that you are transferring, or QMF will replace the existing object with the object that you transfer.

To transfer QMF objects from one user to another, run the following statements:

```

UPDATE Q.OBJECT_DIRECTORY      UPDATE Q.OBJECT_REMARKS      UPDATE Q.OBJECT_DATA
SET OWNER = 'newuserid'        SET OWNER = 'newuserid'        SET OWNER = 'newuserid'
WHERE OWNER = 'olduserid'      WHERE OWNER = 'olduserid'      WHERE OWNER =

```

```
'olduserid'  
AND NAME IN namelist           AND NAME IN namelist           AND NAME IN namelist
```

In these statements, *namelist* is a list of the object names to be transferred; the list must be set off by parentheses, with each name separated by a comma and surrounded by single quotes. For example:

```
('QUERY1', 'QUERY2', 'FORMA', 'PROCB')
```

For queries or procedures that name objects qualified with the old SQL authorization ID, be sure to change the qualifier. For example, if you transfer MYQUERY from BAXTER to JONES, change the name from BAXTER.MYQUERY to JONES.MYQUERY.

Use an SQL statement like the one in “Displaying QMF objects” on page 208 to change the RESTRICTED column value to N if you decide you want to share the object after transferring it.

**Tip:** A single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command.

## Deleting obsolete QMF objects

You must run three statements to delete all of a particular user's QMF queries, forms, procedures, folders, and analytic objects. Make sure that you run all three statements, because the internal representation of each object spans the three QMF control tables (Q.OBJECT\_DIRECTORY, Q.OBJECT\_DATA, and Q.OBJECT\_REMARKS).

```
DELETE FROM Q.OBJECT_DIRECTORY WHERE OWNER = 'olduserid'   DELETE FROM Q.OBJECT_REMARKS WHERE OWNER = 'olduserid'   DELETE FROM Q.OBJECT_DATA WHERE OWNER = 'olduserid'
```

Figure 35. Deleting unnecessary objects from the QMF object control tables

You can also delete obsolete objects by sorting the Q.OBJECT\_DIRECTORY table by date and time. You can select every object where the date last used was before a particular date and delete all the appropriate rows from the three control tables.

**Tip:** A single SQL query can contain multiple SQL maintenance statements. To create a query that includes multiple statements, place a semicolon between statements and set the DSQEC\_RUN\_MQ global variable to 1. Information about how to create queries that include multiple statements as well as restrictions on the types of statements that can be used is included in the description of the RUN command.

## Importing queries, forms, and procedures from z/OS data sets

QMF objects that exist in z/OS data sets can be imported into QMF using the IMPORT command.

If the exported objects are protected by RACF, you need RACF read access to import objects from them. To obtain this access, see your RACF administrator.

## Maintaining a Db2 for z/OS subsystem

If you want to share the responsibility for maintenance, you can assign specialized administration tasks to users to perform under their own authorization IDs.

### About this task

Give these users only enough Db2 authority to run the queries and utilities that are required for their tasks. For example:

- The INSERT privilege is required on the Q.PROFILES table to create QMF profiles for new users.

- DBADM or equivalent authority is required on databases to administer the associated tables, indexes, and table spaces.

Include the WITH GRANT OPTION clause to enable the grantee to, in turn, grant lesser privileges on the database to other users.

- STATS and REORG privileges are required on the database in which the QMF object control tables are stored so that these tables can be monitored and reorganized.

Except where noted, the information relates to Db2 for z/OS. On z/OS, you can maintain multiple databases with multiple table spaces. On Db2 for Linux, UNIX, and Windows, each server (a named location) is a single database. You can maintain multiple table spaces in each database.

To prevent users from accessing QMF during maintenance work, issue the Db2 -STOP DATABASE command for any table spaces that you will be working on before you begin maintenance operations.

## Managing data sets

The data sets for the table spaces and indexes might be user-managed or managed by Db2. How these data sets are managed determines what you must do to enlarge table spaces and indexes.

### Storage groups for data sets managed by Db2

A storage group is a named set of DASD volumes from which space can be drawn for the objects that the storage group supports. For each control table with an index, the index and the table space share a common storage group. The storage group for the QMF control tables is DSQSGCTL. The QMF installation process creates this storage group; do not change this storage group to user-managed from one managed by Db2 after the QMF installation.

If you are using Db2 for Linux, UNIX, and Windows, storage groups do not apply.

### VSAM clusters for user-managed data sets

You need a VSAM cluster for each table space and each index to manage the control-table data sets. You define these clusters using VSAM statements, and link the resulting clusters to Db2 with SQL CREATE statements. The link between a cluster and its Db2 object is in the name of the cluster and the name of the ICF (Integrated Catalog Facility) in which the cluster is cataloged.

### Related reference

[QMF control tables and table spaces for TSO and CICS](#)

These are the control tables shipped with QMF.

## Maintaining the QMF control tables

Most control-table maintenance cannot be done under QMF, because QMF relies on these tables for its operations.

### About this task

You can issue SQL maintenance statements in batch-mode TSO through the DSN processor, or interactively through the SPUFI facility of DB2I. For Db2 for Linux, UNIX, and Windows servers, you can use the Db2 command line processor from the operating system in which the database is installed.

### Monitoring and reorganizing the QMF control tables

You can prevent maintenance problems by monitoring the condition of the control tables through the Db2 system catalog.

### About this task

At its most effective, reorganization can minimize the space requirements for the control tables and indexes and increase the efficiency of QMF operations.



## **Procedure**

1. Periodically run the RUNSTATS utility on the control tables and indexes to add current statistics to certain Db2 system tables.
2. Query these system tables and examine these statistics to decide whether reorganization is required.
3. If reorganization is required, do the following steps:
  - a) Run the REORG utility.
  - b) Rerun the RUNSTATS utility.
  - c) Query the updated system tables again to see if reorganizing improved the statistics.

## **What to do next**

Rebind your most critical applications, such as the QMF application plan, after reorganization so that the most efficient search paths can be selected.

## **Switching buffer pools**

For performance reasons, you might want to change the buffer pool for a table space containing a QMF control table or for a control table index.

For example, you could switch the buffer pools for the control table indexes and table spaces to BP1, and reserve BP1 for their exclusive use.

You change buffer pools through ALTER TABLESPACE and ALTER INDEX statements. You can choose BP0, BP1, or BP2 for your new buffer pool, but not BP32K.



---

## Chapter 13. Setting up printing and charting functions

QMF end users frequently need to print data they retrieve from the database. This data might be in the format of a report, chart, database table, or some other QMF or database object.

How you set up printing for your end users depends on what type of printer you have and which objects you need to print. This topic helps you decide whether it is more efficient for you to handle printing using QMF services or Graphical Data Display Manager (GDDM) services. It also provides instructions on how to print objects using either method.

### Related tasks

[Printing DBCS data from non-DBCS display devices](#)

If you use the Uppercase, Japanese, or Korean National Language Feature (NLF), you might need to print double-byte character set (DBCS) data.

---

### Deciding whether to use QMF or GDDM services for printing

Whether you print using GDDM services or QMF services depends on what type of objects you need to print and what types of printers and other resources are available to you.

Use this topic to help you decide which method suits your needs.

- If you need to print charts, forms, or prompted queries, use GDDM.

QMF uses GDDM services to display these objects; GDDM must be used to print these objects as well. If you do not use GDDM services, you can print only reports, tables, QBE and SQL queries, procedures, and the QMF profile.

- If your site is set up to route output to named printers, use GDDM services for printing.

GDDM allows you to link a name with a device. If you do not use GDDM and use exclusively QMF services, you need to print objects by specifying the type and name of the storage queue through which those objects are routed to the printer.

Both QMF and GDDM handle printer input asynchronously, which means that QMF can return messages indicating that the object is printed before it is actually printed.

If you are using CICS, also consider the following:

- In CICS, if you need to handle routing automatically (rather than writing a program to route output), use GDDM or define transient data queues for use with QMF.

GDDM does the routing for you by using the transient data queue definitions that you define to CICS. QMF takes care of the routing in the same way if you are using transient data queues to hold your output.

If you print to temporary storage, you must write a program to send the temporary storage queue to the printer or display the printed output online with the transaction CEBR supplid with CICS.

- In CICS, if you expect print output to exceed the maximum size of a temporary storage queue, use GDDM for printing or define transient data queues to use with QMF.

## Using GDDM services to handle printing

---

You can use GDDM (rather than QMF) services to handle printing in native z/OS batch, TSO, ISPF, and CICS.

### About this task

**Important:** The explanations in this topic apply only if you are using the GDDM default values shipped with the GDDM product.

### Related information

See the GDDM documentation.

## How QMF interfaces with your GDDM nickname

QMF interfaces with GDDM nicknames through the standard interface provided by GDDM, which issues a call that allows QMF to open a GDDM print file.

The following defaults are provided by QMF on the DSOPEN call when the PRINT command begins:

- The device type is set to Family 2
- The device token is set to \*
- No processing options are in place (PROCOPT is set to zero)
- The only entry in the name list is the nickname

The print operation is carried out one page at a time using the ASCPUT and FSRCE GDDM services. When printing is complete, QMF closes the print operation with a DSDROP statement.

## Where GDDM searches for the nickname

When you enter a printer name on the PRINT command, GDDM searches a single module in CICS, and searches several data sets native in z/OS batch, TSO, and ISPF.

Printer nicknames enable you to define complex print or display devices to simplify the work of your end users. Nicknames define device characteristics that indicate to GDDM how to format and distribute the report, and they can define both local and remote devices.

### Native z/OS batch, TSO, and ISPF

In native z/OS batch, TSO, and ISPF, when a user enters a printer name on the PRINTER keyword of the QMF PRINT command, GDDM first searches the ADMDEFS data set and then the defaults module, ADMADFT, for a matching nickname that defines how and where to direct the output.

### CICS

In CICS, GDDM searches only the defaults module, ADMADFC. GDDM uses nicknames to recognize all the devices with which it can communicate (including display devices).

## Example nicknames for different printer families

These examples show how to define GDDM printer nicknames for different printer families.

### Example nickname for a Family 1 or 2 GDDM printer

To define the nickname GRAPHIC for a Family 1 or 2 GDDM printer, you might use an ADMMNICK specification similar to the one in the following example. This specification is for a Family 2 GDDM printer (use TOFAM=1 for a Family 1 GDDM printer). It uses the device token R87S, an example of a token for a remotely attached 3287 printer.

```
ADMMNICK NAME=GRAPHIC,TOFAM=2,DEVTOK=R87S,TONAME=GRAP
```

- Native z/OS batch, TSO, and ISPF: After you create your nickname in TSO, ISPF, and native z/OS batch, a temporary data set is created as a result of running the QMF PRINT command and specifying a nickname that exists. This data set is `userid.ADMPRINT.REQUEST.#nnnnn`, where `nnnnn` is a sequence number. You can then print the data set by using the ADMOPUT utility. You can also use the ADMOPUJ utility to write your print job to the JES spool.
- CICS: If you use either of the GDDM print utilities (ADMOPUT or ADMOPUJ) to print QMF objects by using GDDM nicknames, the GDDM map groups that are supplied with QMF must be made available to the GDDM print utility. The ADMGGMAP DD statement contains the name of the data set (QMF1210.SDSQMAPE) that holds the map groups:

```
//ADMGGMAP DD DSN=QMF1210.SDSQMAPE,DISP=SHR
```

Without this statement, any attempt to print a form on a Family 2 printer ends in an error.

**Restriction:** In CICS, after you create the ADMMNICK specification, link the name with a device by defining the printer to CICS. Make sure that the TERMINAL attribute in the CICS RDO TERMINAL definition and the TONAME parameter in the ADMMNICK specification have matching values. If you define the printer to CICS by using a DFHTCT macro, make sure that the TONAME parameter in the ADMMNICK specification and the TRMIDNT attribute in the terminal control table (TCT) have matching values.

### Example nickname for a Family 3 GDDM printer

To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one in the following example:

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTK=S3800W8,TONAME=370P
```

After you create your nickname in TSO, ISPF, and native z/OS batch, the ddname ADMLIST is created. You can then send the formatted file to the printer you chose.

After you create the ADMMNICK specification in CICS, link the name with a device by defining the transient data queue to CICS. Make sure that the TONAME parameter in the ADMMNICK specification and the TDQUEUE attribute of the CICS RDO TDQUEUE definition have matching values. If you define the transient data queue by using the DFHDCT macro, make sure that the TONAME parameter in the ADMMNICK specification and the DESTID attribute in the DCT have matching values.

### Example nickname for a Family 4 GDDM printer on TSO, ISPF, or native z/OS batch

To define the nickname 3900PRNT for a Family 4 GDDM printer, you might use an ADMMNICK specification similar to the following one:

```
ADMMNICK NAME=3900PRNT,TOFAM=4,DEVTK=A3820Q
```

Output is written by default to a ddname of ADMIMAGE. You can spool the file to z/OS automatically through JES if you have the CSPPOOL processing option set. Family 4 printers cannot be used in CICS.

#### Related concepts

[Example nickname definitions for specific printers](#)

These examples show printer nicknames that you might use for Family 1, 2, or 3 devices.

#### Related information

Search for information about valid device tokens in the GDDM documentation.

## Example nickname definitions for specific printers

These examples show printer nicknames that you might use for Family 1, 2, or 3 devices.

### 3800, 3812, or 3820 printer, 8 lines per inch:

Use the following definition to define the nickname GDDMPRT2 for a Family 3 printer:

```
GDDMPRT2 ADMMNICK TOFAM=3,DEVTK=S3800N8,NAME=MYPRINT2
```

### Non-3800 system printer, 132 columns, 8 lines per inch:

Use the following definition to define the nickname GDDMPRT3 for a Family 3 printer:

```
GDDMPRT3 ADMMNICK TOFAM=3,DEVTOK=S1403W8,NAME=MYPRINT3
```

### A remotely attached 3287 (suitable for printing charts):

Use the following definition to define the nickname GDDMPRT4 for a Family 2 printer:

```
GDDMPRT4 ADMMNICK TOFAM=2,DEVTOK=R87,NAME=MYPRINT4
```

### Any destination without print control options:

Use the following definition to define the nickname GDDMPRT5 for a Family 3 printer:

```
GDDMPRT5 ADMMNICK TOFAM=3,PROCOPT=((PRINTCTL,0)),NAME=MYPRINT5
```

The PROCOPT parameter specifies processing options by using a print control (PRINTCTL) keyword, with which you specify a number of print control options. For example, you can use PRINTCTL to specify a page heading to be printed, the number of copies to print, and the width of the margins.



**Attention:** If the print data set has RECFM=F, GDDM printing changes the DCB of the data set from RECFM=F to RECFM=V.

### A PC printer using GDDM-PCLK:

Use the following definition to define the nickname PCPRINT for a Family 1 printer:

```
GDDMPRT6 ADMMNICK TOFAM=1,FAM=0,NAME=PCPRINT,TONAME=*,ADMPCPRT
```

In the preceding command, \* indicates the user's current device or the default value.

To print to a workstation printer, GDDM-PCLK must be installed on your workstation.

### Related concepts

[Example nicknames for different printer families](#)

These examples show how to define GDDM printer nicknames for different printer families.

### Related information

Search for information about nicknames for Family 4 printers and about print control options in the GDDM documentation.

## Setting up GDDM services to handle printing

You can use GDDM services to handle printing in native z/OS batch, TSO, ISPF, and CICS.

### Procedure

To use GDDM services to handle printing, complete the following steps:

1. Choose a GDDM nickname for the print device.
2. [“Choosing the right type of GDDM device” on page 217.](#)
3. [“Creating the printer nickname specification” on page 217.](#)
4. [“Activating the nickname” on page 219.](#)
5. [“Updating the GDDM defaults module with the nickname” on page 218.](#)
6. Update the PRINTER field of the user's row in the Q.PROFILES table.

### Related concepts

[Where GDDM searches for the nickname](#)

When you enter a printer name on the PRINT command, GDDM searches a single module in CICS, and searches several data sets native in z/OS batch, TSO, and ISPF.

### Choosing the right type of GDDM device

The printer nickname you use depends on the type of device.

#### Family 1 devices

Specify auxiliary devices attached to a workstation using GDDM-PCLK. A Family 1 device can also include display devices, such as 3270 data-stream display devices.

#### Family 2 devices

Include devices such as IBM 3270 display devices and queued printers.

#### Family 3 devices

Are system printers that support the ANSI code of carriage control characters.

#### Family 4 devices

Are printers for which you need to use the ADMOPUT and ADMOPUJ utilities to print output. These utilities are provided by GDDM. Family 4 devices cannot be used for printing in CICS.

### Creating the printer nickname specification

Use this format to specify a GDDM printer nickname.

---

```
ADMMNICK NAME=nickname,TOFAM=family_type,DEVTOK=device_token,TONAME=name
```

---

Figure 36. Format of the ADMMNICK specification, which defines a printer nickname

- 
- Use NAME to indicate a 1-character to 8-character printer nickname to use with the QMF PRINT command. For example, if MYPRTR is the nickname, users can enter the following command: PRINT REPORT (PRINTER=MYPRTR. NAME can be a single name, a list of names separated by commas, or a name with a leading or trailing ? character, which is used as a wildcard to send output to multiple printers that have similar names.
  - Use TOFAM to indicate the type of device you are using. GDDM recognizes four families of devices, and handles each differently.
  - Use DEVTOK to indicate a valid GDDM device token, which uniquely identifies a device and its print configuration.
  - TONAME is used only in CICS. This field points to a CICS terminal definition or a CICS transient data queue definition, which enables CICS to properly manage communication between GDDM and the printer. Use the TONAME parameter to point to the name of a 1-character to 4-character CICS definition with a value that depends on the type of device:
    - If the nickname defines a Family 1 or 2 printer, the TONAME parameter must point to a corresponding CICS terminal definition.

If you define the printer to CICS using CICS resource definition online (RDO) to update the CICS system definition (CSD) file, the TERMINAL attribute has the same value as TONAME.

If you define the printer to CICS using a DFHTCT macro, which creates an entry in the CICS terminal control table (TCT), the TRMIDNT field has the same value as TONAME.
    - If the nickname defines a Family 3 printer, the TONAME parameter must point to a corresponding CICS transient data queue.

If you define the transient data queue to CICS using CICS resource definition online (RDO) to update the CICS system definition (CSD) file, the TDQUEUE attribute has the same value as TONAME.

If you define the transient data queue to CICS using the DFHDCT macro, which creates a corresponding entry in the CICS destination control table (DCT), the DESTID field has the same value as TONAME.

## Adding the nickname in native z/OS batch, TSO, and ISPF

Add the nickname to your ADMDEFS data set. GDDM looks at this data set first. If the nickname is not found, GDDM looks in the external defaults module, ADMADFT, in which you define a GDDM ADMMNICK specification. See [Figure 36 on page 217](#) for the proper format of the ADMMNICK specification.

## Adding the nickname in CICS

To create a nickname in CICS, first define a GDDM ADMMNICK specification, as shown in [Figure 36 on page 217](#), in the GDDM external defaults module, ADMADFC. This specification indicates the device characteristics to GDDM, such as the number of lines per page the printer can handle, and how the printer is managed by CICS.

A unique label can be added to the syntax. For example, GDDMPRT1 is a possible label for the nickname definition in the following example:

```
GDDMPRT1 ADMMNICK NAME=MYPRINT,TOFAM=3,DEVTOK=ADMKSYSP
```

## Defining multiple nicknames with one definition

You can use a single nickname to define multiple printer addresses by including the wildcard ? in your nickname definition, like this:

```
ADMMNICK TOFAM=3,NAME=MYPRINT?,PROCOPT=((PRINTCTL,0))
```

The nickname MYPRINT? allows you to route print output to printers named MYPRINT1, MYPRINT2, MYPRINTA, and so on. For example:

```
PRINT REPORT (PRINTER=MYPRINT2
```

In this example, GDDM uses the nickname definition for the MYPRINT? nickname to create a data set and direct the output from the PRINT command to the data set with a ddname of MYPRINT2.

## Related concepts

[Example nickname definitions for specific printers](#)

These examples show printer nicknames that you might use for Family 1, 2, or 3 devices.

[Example nicknames for different printer families](#)

These examples show how to define GDDM printer nicknames for different printer families.

## Related information

Search for information about the GDDM print utilities and about Family 4 printing in the GDDM documentation.

## Updating the GDDM defaults module with the nickname

After you create the printer nickname specification, update the GDDM defaults module ADMADFC or ADMADFT.

## About this task

The defaults module also contains default values for the GDDM product. The module is stored as a member of the SADMSAM data set.

## Procedure

To update the module with your nickname specification:

1. Edit the source file to add the nickname.
  - In native z/OS batch,TSO, and ISPF, the external defaults module is ADMADFT.
  - In CICS, the external defaults module is ADMADFC.
2. Enter your ADMMNICK specification after the ADMMDFT statements in the module.
3. Reassemble and link-edit the changed defaults module.



## Related information

Search for information about the defaults module in the GDDM documentation.

## Activating the nickname

After you updated the GDDM defaults module with the nickname specification, you need to test and incorporate the nickname so that it can be used. This process differs depending on your operating environment.

## Native z/OS batch, TSO, and ISPF

Test your nickname definitions by placing them in an external defaults file and printing with them until you are satisfied they are working correctly. Then, you can assemble them into external defaults modules.

GDDM uses external defaults modules more efficiently than data sets to find a nickname.

The decision to use external defaults files or modules affects a user's JCL because an external defaults file requires a DD statement, while an external defaults module must be a member of a STEPLIB library.

For TSO, ISPF, native z/OS batch, the ddname of the nickname data set is ADMDEFS. You should allocate it when you start your QMF session. To add the ddname ADMDEFS to the user's logon procedure, use a statement like the following one:

```
//ADMDEFS DD DSN=LOCAL.GDDM.NICKNAME,DISP=SHR
```

## CICS

In CICS, the nicknames are incorporated into user default specifications and assembled into the external defaults module, ADMADFC.

After you update the ADMADFC module, you need to update the CICS resource definitions so that CICS can link the nickname with the device it manages.

- Linking a Family 2 nickname with a device: QMF supports the use of GDDM nicknames for reports and requires nicknames for printing QMF charts, forms, and prompted queries. If you have printers that are described to CICS by using VTAM and CICS terminal definitions, you must describe the printer as **queued**, an attribute of GDDM Family 2 devices. When you use a Family 2 device, the TONAME parameter in your ADMMNICK specification points to a CICS terminal definition (as opposed to a CICS transient data queue definition for a Family 3 device).

For example, consider the following nickname specification:

```
ADMMNICK NAME=GRAPHIC,TOFAM=2,DEVTOK=R87S,TONAME=GRAP
```

For this nickname specification, you can update the CICS TCT by using a macro similar to the following example.

```
GRAP      DFHTCT TYPE=TERMINAL,
           ACCMETH=VTAM,
           TRMIDNT=GRAP,
           TRMTYPE=SCSPRT,
           . . .
           . . .
```

Figure 37. Defining to CICS a nickname for a Family 2 GDDM printer

To define the printer to CICS by using RDO, follow this example:

```
DEFINE TERMINAL(GRAP) TYPETERM(DFHSCSP) ...
```

- Linking a Family 3 nickname with a device: To use Family 3 devices, set up a GDDM nickname table by using the following syntax. See [Figure 36 on page 217](#) for the meanings of the parameters that are shown in the ADMMNICK specification.

```
GDDMPRT  ADMMNICK  TOFAM=3,           X
           NAME=SYSPRT,                X
           DEVTOK=S1403W6,             X
           TONAME=SYSP
```

*Figure 38. Defining to CICS a nickname for a Family 3 GDDM printer*

The GDDM documentation (which you can find in the IBM Publications Center) describes the process of incorporating the nicknames into the user default specifications and assembling the user default specifications into the external defaults module, ADMADFC.

The TONAME parameter must have a matching CICS transient data queue definition. You can define the transient data queue to CICS by using RDO as shown here:

```
DEFINE TDQUEUE(SYSP) TYPE(EXTRA) BLOCKSIZE(6050) DDNAME(ADMSYSP)
RECORDFORMAT(VARIABLE) BLOCKFORMAT(BLOCKED)
RECORDSIZE(260) TYPEFILE(OUTPUT) . . .
```

To define the transient data queue to CICS as a DCT entry, see [Figure 39 on page 220](#).

```
* THE GDDM NICKNAME IS SYSPRT AND THE
* LONGEST RECORD THAT CAN BE PRINTED
* IS 256.

      DFHDCT TYPE=SDSCI, DSCNAME=ADMSYSP,           X
      RECFORM=VARBLK,                               X
      RECSIZE=260, BLKSIZE=6050, TYPEFLE=OUTPUT

      .
* ENTRY FOR GDDM NICKNAME SYSPRT
SYSP      DFHDCT TYPE=EXTRA, DESTID=SYSP, DSCNAME=ADMSYSP, RSL=1
```

*Figure 39. Adding a TONAME entry to the CICS DCT*

You also need to add the ddname ADMSYSP to the CICS startup JCL, as follows:

```
//ADMSYSP DD SYSOUT=A
```

Add the TYPE=SDSCI entry that is shown in [Figure 39 on page 220](#) after all other TYPE=SDSCI entries in the DCT.

## Using QMF services to handle printing

You can use QMF (rather than GDDM) services to handle printing in native z/OS batch, TSO, ISPF, and CICS.

### QMF services for printing in native z/OS batch, TSO, and ISPF

You can use DSQPRINT to print a report, table, SQL or QBE query, procedure, or your profile.

DSQPRINT is a special printer destination that QMF uses when you do not supply a printer name on the command line or in the user profile. DSQPRINT must be allocated with a DD statement that points to the data set or output class QMF uses for printing. The DD statement becomes part of your QMF startup exec, CLIST or JCL.

To add your printed output to a user-owned data set, allocate DSQPRINT by using JCL or a CLIST. Here is an example of JCL to allocate DSQPRINT:

```
//DSQPRINT DD DSN=&SYSUID..PRINT.DATA,DISP=MOD
```

Here is an example of a CLIST that accomplishes the same thing:

```
ALLOC DDNAME(DSQPRINT) SYSOUT(A) LRECL(133) RECFM(F B A) BLKSIZE(1330)  
FREE DDNAME(DSQPRINT)
```

To route your output to a printer, allocate DSQPRINT by using the following syntax:

```
//DSQPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
```

Set the carriage control (CC) option to YES when you issue the PRINT command.

If you are using ISPF, you can use the DPRE (Display Printed Report) command synonym that is supplied with QMF to view the effects of the width and length values you specified without having to print the report. This view is applicable only while you use DSQPRINT.

### Related concepts

The DPRE synonym: using ISPF to preview the printed report

IBM supplies the QMF command synonym DPRE to display a report as it appears when printed. When you issue the DPRE command, QMF completes the report and prints the report to a data set that is allocated to DSQPRINT. The ISPF browser is then called to view this data set.

## Using QMF services for printing in CICS

To use QMF services to handle printing, specify the type of storage you want to use and provide CICS with a name for the storage.

### Choosing between temporary storage queues and transient data queues

CICS temporary storage queues have a maximum size and route data only to local print destinations. If you need to route data to other print destinations or the print output is too large for a temporary storage queue, you need to write a program that routes the data from the temporary storage queue to a transient data queue, or view the report online with the transaction CEBR, supplied with CICS.

CICS transient data queues are limited only by the amount of storage defined for the queue before CICS is started. You can define the transient data queue as an intrapartition or extrapartition data queue. You can use transient data queues to print data to a data set or SYSOUT class.

### Using the PRINT command to route output to queues

You can specify on the QMF PRINT command both the name of the queue and the type of storage defined for that queue. For example, to print a report to a temporary storage queue named XYZ, enter this command:

```
PRINT REPORT (QUEUET=TS,QUEUEN=XYZ)
```

To print from a transient data queue named XYZ, you can enter the following command. Ensure that the transient data queue is defined to CICS before its first use.

```
PRINT REPORT (QUEUET=TD,QUEUEN=XYZ)
```

QUEUET and QUEUEN are abbreviations for QUEUETYPE and QUEUENAME.

QMF issues an ENQ statement on the queue name to prevent writing to the queue if another program is using it. If the name is already enqueued by another application, CICS indicates to QMF that the queue is unavailable at that time. Use the SUSPEND (S) keyword to tell QMF what to do when the queue is unavailable. Use the value YES (or Y) to hold the report until the queue is available, then write to it. For example:

```
PRINT REPORT (QUEUET=TS,QUEUEN=XYZ,S=YES)
```

The value NO is the default and cancels the PRINT command, returning a message to the user.

## Using global variables to define queues for printing

If you do not specify values for queue type and queue name on the PRINT command, QMF uses values stored in the global variables DSQAP\_CICS\_PQTYPE and DSQAP\_CICS\_PQNAME.

Set the global variable DSQAP\_CICS\_PQTYPE to TS if you are using temporary storage queues for printing, and TD if you are using transient data queues. TS is the default.

Use the global variable DSQAP\_CICS\_PQNAME to define the name of the temporary storage or transient data queue. Names for transient data queues can be from 1 to 4 bytes. Names for temporary storage queues can be from 1 to 8 bytes. The default temporary storage queue name is DSQPnnnn, where nnnn is the user's 4-byte CICS terminal ID. For example, DSQPA085 is a valid name.

## Printing from a CICS temporary storage queue

If you set up your environment to route print output to temporary storage queues, you need to write a transaction that routes the output from the queue to a printer. The QMF user can then start the print transaction by using the CICS command. Any subsequent PRINT command from the same device uses the same queue name, appending the previous report. See for more information about the CICS command.

## Viewing a report from a CICS temporary storage queue

You can view a report with the transaction CEBR, supplied with CICS.

## Allowing users to print without exiting QMF

You can customize your system to allow users to use a local print utility by pressing the Print function key to print an object without exiting QMF. You define a command synonym for printing and customize your Print function key.

### Procedure

#### 1. Create a file to print the current object locally:

- If you are running in native z/OS batch, TSO, or ISPF

Create a REXX exec or CLIST to locally print the current object. Here is a sample that uses a QMF callable interface application named PRTQMF:

```
/* PRTQMF REXX EXEC for local DSPRINT */
CALL DSQCIX "PRINT PROC (PRINTER=MYPRINT1"
DSPRINT '&SYSUID..MYPRINT1.DATA'
```

This example assumes you have a MYPRINT1 nickname defined and that it directs print output to a data set called MYPRINT1.DATA.

Some users prefer to bypass the PRINT command and simply export the object for local printing. In this case use something like the following exec:

```
/* PRTQMF REXX EXEC for local print utilities called DSPRINT */
CALL DSQCIX "EXPORT PROC TO MYPROC"
DSPRINT '&SYSUID..MYPROC.PROC'
```

- If you are running in CICS

Create a QMF procedure called PRT\_QMF. This sends the object to temporary storage, then starts a transaction that prints the object.

```
PRINT REPORT (QUEUE=QMFREPT,QUEUETYPE=TS)
CICS QMFP (FROM='QMFREPT')
```

#### 2. Create a QMF command synonym for printing.

These sample queries create the command synonym PRTQMF to execute the PRTQMF program:

- If you are running in native z/OS batch, TSO, or ISPF

Create a command synonym like this sample:

```
INSERT INTO COMMAND_SYNONYMS (VERB, SYNONYM_DEFINITION, REMARKS)
VALUES('PRTQMF','TSO PRTQMF','Print QMF Proc')
```

- If you are running in CICS

Create a command synonym like this sample:

```
INSERT INTO COMMAND_SYNONYMS (VERB, SYNONYM_DEFINITION, REMARKS)
VALUES('PRTQMF','RUN PRT_QMF','Print QMF Report')
```

### 3. Customize a function key to use this command synonym.

You must customize a key for each panel.

These examples assume that the user's profile has the PFKEYS column value set to PFKY\_TABLE:

- If you are running in native z/OS batch, TSO, or ISPF

Customize a function key on the **PROC** panel. This example shows a query to customize function key 4 on the procedure panel:

```
INSERT INTO PFKY_TABLE (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('PROC','K',4,'PRTQMF')
```

- If you are running in CICS

Customize a function key on the report panel. A query to customize function key 4 on the **REPORT** panel looks like this:

```
INSERT INTO PFKY_TABLE (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('REPORT','K',4,'PRTQMF')
```

### 4. Restart QMF to implement the function key change.

#### Related tasks

##### Customizing Command synonyms

If the default command synonyms do not meet your needs, you can create your own synonyms. After you create a command synonym, users can enter the synonym on the command line in the same way that they enter a QMF command.

##### Customizing QMF function keys

You customize the command that QMF issues when a user presses a function key and the key label in the function key table.

## Printing requirements by object type

The rules for printing QMF and database objects vary, depending on the type of object.

The following table summarizes the requirements for each object.

| <i>Table 49. Summary of print requirements for QMF and database objects</i> |                          |                                                                |                                        |
|-----------------------------------------------------------------------------|--------------------------|----------------------------------------------------------------|----------------------------------------|
| <b>Object type</b>                                                          | <b>Nickname required</b> | <b>When GDDM gets control</b>                                  | <b>Where output is routed for z/OS</b> |
| Chart                                                                       | Yes                      | GDDM ICU always gets control when the PRINT command is issued. | Output is controlled by GDDM.          |

Table 49. Summary of print requirements for QMF and database objects (continued)

| Object type    | Nickname required | When GDDM gets control                                                                     | Where output is routed for z/OS                                                     |
|----------------|-------------------|--------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Form           | Yes               | GDDM always gets control when the PRINT command is issued.                                 | Output is controlled by GDDM.                                                       |
| QBE query      | No                | GDDM gets control only if the nickname is supplied on the PRINT command or in the profile. | Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT. |
| Procedure      | No                | GDDM gets control only if the nickname is supplied on the PRINT command or in the profile. | Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT. |
| Profile        | No                | GDDM gets control only if the nickname is supplied on the PRINT command or in the profile. | Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT. |
| Prompted query | Yes               | GDDM always gets control when the PRINT command is issued.                                 | Output is controlled by GDDM.                                                       |
| Report         | No                | GDDM gets control only if the nickname is supplied on PRINT command or in the profile.     | Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT. |
| SQL query      | No                | GDDM gets control only if the nickname is supplied on the PRINT command or in the profile. | Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT. |
| Table          | No                | GDDM gets control only if the nickname is supplied on the PRINT command or in the profile. | Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT. |

## Enabling charting functions

QMF creates charts using the Interactive Chart Utility (ICU) supplied by the GDDM-PGF product.

### About this task

From a single report, users can specify different chart formats, such as scatter charts, pie charts, and bar charts. When the user enters a QMF DISPLAY CHART or EXPORT CHART command, the chart format and the data are merged to produce graphics data file (GDF) data. Users can use IBM-supplied chart formats or create their own. In addition, they can save newly created chart formats, provided they have libraries in which to store them.

### Related concepts

[Installation prerequisites for requester \(Db2 for z/OS\) databases](#)

Before you can install QMF on Db2 for z/OS databases that function as stand-alone or requester databases, you must fulfill the hardware and software requirements.

## Enabling chart support in TSO and ISPF

From a single report, users can specify different chart formats, such as scatter charts, pie charts, and bar charts.

### About this task

Users can use default chart formats or create their own. In addition, they can save newly created chart formats if they have libraries in which to store them.

### Procedure

To create a library to hold a user's saved chart formats:

1. Create the library with a DD statement:

```
//DSQUCFRM DD DSN=aaaaaaaa,DISP=(NEW,CATLG),  
//          UNIT=xxxx,VOL=SER=yyyy,  
//          SPACE=(400,(200,50,25)),  
//          DCB=(LRECL=400,BLKSIZE=400,RECFM=F)
```

Provide the DSN, UNIT, VOL, and SPACE parameters, but do not change the DCB parameters.

2. Allocate the library for the user's QMF sessions, and use the ddname DSQUCFRM.

You can allocate the data set through the user's TSO logon procedure, or you can allocate it through a CLIST that the user calls to reach QMF.

For example:

```
ALLOC DSNAME(aaaaaaaa) DDNAME(DSQUCFRM) SHR
```

### What to do next

The default chart formats are in the library QMF1210.SDSQCHRT by default. Allocate this library to the ddname ADMCFORM. Both this library and the user's library are searched for user-specified chart formats, but the new library is searched first. When the user saves a chart format, it always goes into the new library, never into QMF1210.SDSQCHRT.

This arrangement gives each user access to both the default chart formats and formats that the user saved. It also prevents replacement of the default chart formats.

## Enabling chart support in CICS

During QMF installation, a data set is created to hold the default chart formats. This data set is described to CICS by a CSD file entry with the name DSQUCFRM.

This data set is normally allocated to the CICS region during CICS startup and is available to all CICS users. The DSQUCFRM data set is the default chart library used to store chart formats when using the ICU from QMF. You can store chart formats in other chart libraries by using the advanced form of the ICU panel directory. Each chart library must be described to CICS and must be accessible to the CICS region that is running QMF. You describe the chart library with a file entry in the CSD data set.

QMF provides an EXPORT CHART command. This command is used to save the whole chart in graphic data format (GDF). When you export a chart, the GDF data is stored in the GDDM ADMF library. You can also save the whole chart in GDF using the ICU facility of GDDM.

### Related information

Search for information about the advanced ICU panel directory in the GDDM documentation.





---

# Chapter 14. Command synonyms

Command synonyms help you customize QMF commands by allowing you to define your own terms and link them to QMF, CICS, or TSO commands. A synonym can be another word for a command, or it might be a term that does the work of several commands.

---

## Using the default synonyms provided with QMF

---

QMF provides several command synonyms by default.

### List of default synonyms

QMF provides several command synonyms that appear in the Q.COMMAND\_SYNONYMS table. Users with access to this table can call an application associated with one of the default command synonyms by entering the synonym as if it were a QMF command.

If you are connected to Db2 for Linux, UNIX, and Windows, the default synonyms appear in the table Q.COMMAND\_SYN\_TSO.

These default synonyms are valid only if QMF is started under ISPF. If QMF is not started under ISPF, you can access ISPF by entering TSO ISPSTART. ISPF is not available when QMF is running under CICS.

#### **DPRE (display printed report)**

The synonym for the display printed report REXX exec is DPRE. This exec displays the user's current report in its printed form.

#### **BATCH (batch query/procedure)**

The synonym for the batch application is BATCH. This application allows the user to run a query or procedure in batch mode.

#### **ISPF (bridge to ISPF)**

The synonym to bridge to ISPF is ISPF. This synonym lets the user temporarily leave the interactive mode of QMF and "bridge" to an ISPF/PDF session. After the session ends, the user returns to QMF at the point where the ISPF command was issued.

#### **LAYOUT (layout form)**

The synonym for the Layout Form application is LAYOUT. This application lets the user customize reports without running a query.

#### **RUNTSO**

The RUNTSO command makes it possible for QMF for Workstation and QMF for WebSphere users to run queries and procedures using the stored procedure interface to QMF for TSO. When a query or procedure that includes the RUNTSO command is passed to QMF for TSO from the stored procedure interface, QMF for TSO issues the RUN command with any parameters that were originally passed on the RUNTSO command from the client.

Before users can issue the RUNTSO command from either QMF for Workstation or QMF for WebSphere, the stored procedure interface must be installed and configured.

#### **RU**

RU is a synonym for the RUN command. This command synonym is needed if you use RU as an abbreviation for the RUN command.

#### **Related concepts**

The DPRE synonym: using ISPF to preview the printed report

IBM supplies the QMF command synonym DPRE to display a report as it appears when printed. When you issue the DPRE command, QMF completes the report and prints the report to a data set that is allocated to DSQPRINT. The ISPF browser is then called to view this data set.

Running QMF in batch mode

If a user runs a procedure with the RUN command, the user cannot execute QMF commands except to cancel the procedure or the session. Thus, running a procedure using the RUN command can tie up considerable session time.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## The DPRE synonym: using ISPF to preview the printed report

IBM supplies the QMF command synonym DPRE to display a report as it appears when printed. When you issue the DPRE command, QMF completes the report and prints the report to a data set that is allocated to DSQPRINT. The ISPF browser is then called to view this data set.

A printed report does not look exactly as it does on screen. The following differences exist between a displayed report and a printed report:

- A report with one or more page breaks displays as a single scrollable page. A printed report with one or more page breaks contains multiple pages.
- Page headings and footings appear once in a displayed report. In a printed report, they display at the top and bottom of each page.
- Fixed columns remain in place when a displayed report is scrolled horizontally. In a printed report, fixed columns are repeated on the left side of each page.

The synonym definition that is associated with the DPRE command runs a REXX exec named DSQAnR1C (where *n* is a one-character national language identifier that depends on the language in which QMF is running). DSQAnR1C calls an associated exec named DSQABR1C. These applications are shared for use by everyone.

### Using DPRE

To use DPRE, load the DATA object with the report data and the FORM object with the appropriate form.

After you load the objects, issue the following command:

```
DPRE
```

The DPRE application then generates the printer output and displays it through the ISPF browse facility.

If you are using an National Language Feature (NLF), issue the translated command synonym for DPRE to display printed reports. For example, the translated German command synonym for DPRE is AGB. For the translated command synonym for DPRE in the other language environments, see the Q.COMMAND\_SYNONYM\_*n* control table. Replace the *n* character with the 1-character language identifier for your NLF.

### Report parameters

The LENGTH parameter for the report being browsed is taken from the user's profile. The WIDTH parameter specified in the profile is used if it has a logical record length (LRECL) of less than 132; otherwise, a record length of 132 is used because this is the length specified in the TSO allocation statement. If 132 is too small, the TSO allocation statement for DSQPRINT can be changed to accommodate a larger width.

### Responding to errors

DSQPRINT is the ddname for the data set that receives output from QMF PRINT commands in which PRINTER=' ' in cases where QMF for TSO was started under ISPF. When a user runs DPRE, DSQPRINT is redefined as the data set that holds the material to be browsed. If the DPRE command stops for any reason before the report can be displayed, printed output might not go to the expected location. To correct this problem, you must exit QMF and begin a new session.

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### **Customizing DPRE**

Under TSO, you can customize the behavior of the DPRE command synonym by modifying the DSQABR1C exec.

The DSQABR1C exec contains a customization section at the beginning of the program. You can use the customization section to change settings such as the name of the BROWSE data set or the printer allocation. The DSQABR1C exec contains instructions for modifying the customization section.

**Important:** When you modify any file, first rename it and be sure to keep backup copies of the original and modified files.

## Guidelines for synonyms

---

Command synonyms include a verb, a definition, and optionally, an object name.

### **Synonym verbs**

Every command synonym definition must have a verb. Only the object name is optional.

The verb is your own word for the command in QMF for CICS, or QMF for TSO that is stored in the SYNONYM\_DEFINITION column. For example, you might create the synonym COMPUTE for the QMF base verb RUN if your company has financial analysts who run procedures that return computations.

### **Related concepts**

#### Synonym object names

An object name is optional in a command synonym. When you do use an object name, however, ensure that users specify both the verb and the object name; otherwise, QMF cannot find a match in the synonyms table.

#### Synonym definitions

The synonym definition is the command, procedure, or application that runs when the user enters the command synonym.

### **Rules for the VERB column**

Ensure that entries in the VERB column of the synonyms table are valid.

Entries in the VERB column of the synonyms table must follow these rules:

- Are 1 to 18 characters long
- Do not contain blanks
- Do not include the verb QMF (other base QMF commands are allowed)
- Have an alphabetic or national character as the first character

In English, national characters are #, @, and \$.

Characters after the first letter can be alphabetic, national characters, decimal digits, or the underscore. No other characters are allowed.

The following examples demonstrate these rules. QMF ignores rows that have invalid entries in the VERB column.

| <i>Table 50. Examples of valid and invalid verbs for command synonyms</i> |                                                                           |
|---------------------------------------------------------------------------|---------------------------------------------------------------------------|
| <b>Valid verbs</b>                                                        | <b>Invalid verbs</b>                                                      |
| COMPUTE                                                                   | DO SALES (Blanks not allowed unless surrounded by double quotation marks) |
| DISPLAY                                                                   | ADJ%AGE (% not allowed)                                                   |
| PRINT                                                                     | PRINT_PRODUCTIVITY_TOTALS (more than 18 characters)                       |

### Using base QMF verbs as command synonym verbs

You can use base QMF commands, such as PRINT, as synonyms.

For example, you might choose to define a synonym that automatically routes print output to a GDDM-defined printer.

When you define a synonym that is also a base QMF command, you must precede the command with the letters QMF if you want to use the base QMF command. For example, the synonym DISPLAY might represent a synonym definition that executes the QMF command RUN PROC SALES\_REPORT. The SALES\_REPORT procedure runs a query and prints a report on a GDDM-defined printer. Users who forget to enter QMF in front of DISPLAY might get a formatted, printed report of data they did not necessarily want.

Some base QMF commands must be followed by keywords or parameters. For example, you need to follow the IMPORT command with an object type, such as TABLE. If you are using a verb such as IMPORT in a verb-object pair, choose an object name that is not one of these parameters to prevent users from inadvertently running the synonym.

### Synonym object names

An object name is optional in a command synonym. When you do use an object name, however, ensure that users specify both the verb and the object name; otherwise, QMF cannot find a match in the synonyms table.

Entries in the OBJECT column of the synonyms table must follow these rules:

- Must be 1 to 31 characters long
- Must conform to the rules for naming Db2 tables
- Must be surrounded by double quotation marks if the object name has blanks or other special characters

Both QMF and the database manager remove the double quotation marks when the name is processed.

The following examples show valid and invalid objects:

| <i>Table 51. Examples of valid and invalid objects in command synonym definitions</i> |                                                          |
|---------------------------------------------------------------------------------------|----------------------------------------------------------|
| <b>Valid objects</b>                                                                  | <b>Invalid objects</b>                                   |
| PFKEYS                                                                                | 80CAT (first character is numeric)                       |
| MONTH_2_REPORT                                                                        | ADJ%AGE (% not allowed)                                  |
| "Net Sales"                                                                           | JONES GROSS (double quotation marks required for blanks) |

### Related concepts

[Synonym verbs](#)

Every command synonym definition must have a verb. Only the object name is optional.

[Synonym definitions](#)

The synonym definition is the command, procedure, or application that runs when the user enters the command synonym.

## Synonym definitions

The synonym definition is the command, procedure, or application that runs when the user enters the command synonym.

### Related concepts

#### Synonym verbs

Every command synonym definition must have a verb. Only the object name is optional.

#### Synonym object names

An object name is optional in a command synonym. When you do use an object name, however, ensure that users specify both the verb and the object name; otherwise, QMF cannot find a match in the synonyms table.

### Guidelines for definitions

An entry in the SYNONYM\_DEFINITION column can include certain commands.

An entry in the SYNONYM\_DEFINITION column can include:

- A RUN command that calls a QMF procedure or query

For example, the following command might be a synonym definition for the synonym COMPUTE MONTHLY\_SALES:

```
RUN PROC JONES.SALES_DATA
```

- A TSO command that calls a CLIST
- A CICS command that starts another CICS transaction

### Example: using a linear procedure in the synonym definition

A linear procedure is a QMF procedure that executes QMF commands sequentially. Your synonym definition can include a linear procedure that does the work of several QMF commands.

For example, consider the following procedure:

```
-- Procedure name: SALES_PROC  
RUN QUERY SALES_DATA  
PRINT REPORT (QUEUE=XYZ,QUEUETYPE=TS)  
TSO RPTX (FROM=('REPORTX, XYZ'))
```

Figure 40. Sample procedure to run using a command synonym

This procedure performs the following tasks:

1. Runs the following query, called SALES\_DATA, which creates a report that shows all the customers handled by sales representative number 20:

```
SELECT QUANTITY, CUSTNO  
FROM Q.SALES  
WHERE SALESREPNO = 20
```

2. Routes the report from QMF to TSOTSO virtual storage or a CICS temporary storage queue. XYZ is the name of the temporary storage queue.
3. Runs a CICS or TSO procedure to route the report from storage to a predefined print destination. RPTX is the transaction name. It runs asynchronously with QMF to route output to a destination named REPORTX.

Your definition for a synonym that runs this procedure might look similar to the following one:

| SYNONYM<br>VERB | OBJECT | DEFINITION          |
|-----------------|--------|---------------------|
| SHOW            | SALES  | RUN PROC SALES_PROC |

Figure 41. Example of using a command synonym to run a linear procedure

If you are using an National Language Feature (NLF), ensure that the commands in the queries, forms, and other objects included in the procedure are translated before you use the command synonym (unless your procedure sets the DSQEC\_NLFCMD\_LANG variable to 1, which requires that users issue commands in English). Also ensure that these components are suitable for the NLF that you are using.

### Using variables in the synonym definition

You can use variables in the synonym definition to pass values for like-named variables present in objects (such as queries) named in the definition.

For example, this example shows a definition that passes the value Q.STAFF for the table name, which is evaluated when MYQUERY runs.

| SYNONYM<br>VERB | OBJECT | DEFINITION                              |
|-----------------|--------|-----------------------------------------|
| EXECUTE         | -      | RUN QUERY MYQUERY (&&TABLENAME=Q.STAFF) |

Figure 42. Using variables in command synonym definitions

MYQUERY might look something like the following:

```
SELECT * FROM &TABLENAME
```

Ampersands are doubled in a variable name in the synonym definition because they become single ampersands when QMF executes the RUN command.

### The variable &ALL

&ALL is a special QMF variable that allows you to enter variable values when you enter the synonym, rather than including them in the synonym definition. Use double ampersands in the synonym definition for all variables except the variable &ALL.

&ALL is a special QMF variable that allows you to enter variable values when you enter the synonym, rather than including them in the synonym definition. When you use the variable &ALL in a synonym definition, QMF uses as variable values any information you enter to the right of the synonym. You can use the &ALL variable to show where the variable information is located within the synonym definition.

This synonym definition shows an example of a synonym defined using the &ALL variable.

| SYNONYM<br>VERB | OBJECT | DEFINITION                  |
|-----------------|--------|-----------------------------|
| SHOW_INFO       | -      | RUN QUERY STAFFQUERY (&ALL) |

Figure 43. Using the variable &ALL in a command synonym definition

The query named STAFFQUERY might look something like the following example:

```
SELECT * FROM Q.STAFF
WHERE DEPT=&DEPT and JOB=&EMPLOYEE_JOB
```

After activating the SHOW\_INFO synonym defined in Figure 43 on page 232, you can enter the following statement from the QMF command line to display information about all the managers in Department 10:

```
SHOW_INFO &DEPT=10 &EMPLOYEE_JOB='MGR'
```

When you use the variable &ALL in a synonym definition, follow these guidelines:

- Use &ALL only once in a synonym definition.
- Always write &ALL in upper case.
- Never follow &ALL with a number or letter.
- Any value you substitute for &ALL must be syntactically correct when QMF evaluates the entire command.

If a user does not supply a value following the command synonym, QMF substitutes a null value for &ALL. In the synonym definition shown in Figure 43 on page 232, QMF prompts the user for values for the &DEPT and &EMPLOYEE\_JOB variables if the user enters SHOW\_INFO by itself on the command line.

### Keying information into the SYNONYM\_DEFINITION column

Command synonym definitions must follow certain rules when you enter them into the synonyms table.

Follow these rules when keying your synonym definitions into the synonyms table:

- Add single quotes around any variable in your synonym definition.

Single quotes around a variable eliminate the need for the user to add quotes to the command synonym when running a query. For example:

```
RUN MYQUERY (&&NAMEVALUE='&ALL')
```

If you search for the name MARENGHI, you do not need to enter 'MARENGHI', because QMF does this for you.

- Enter base verbs and keywords in upper case.

Literal information in the synonym definition is not converted to upper case.

- Qualify all object names if their owners are different from the SQL authorization ID of the user who uses the synonym.

QMF leaves names unqualified when searching for a synonym that contains the object name specified. For example, if your synonym definition includes a query named MY\_SALES owned by user ID JONES, ensure that the object name in the synonym definition reads JONES.MY\_SALES. Otherwise, JONES is the only user who can use that command synonym.

- Use only capital letters for letters that lie outside of delimited identifiers.

If QMF converts user input (the synonym) to upper case and the synonym definition is in lower case, QMF cannot find the synonym definition that matches the synonym that the user entered. The CASE value of the user's QMF profile controls whether input is converted to upper case. Use the SET PROFILE command to change the CASE value.

## Customizing Command synonyms

---

If the default command synonyms do not meet your needs, you can create your own synonyms. After you create a command synonym, users can enter the synonym on the command line in the same way that they enter a QMF command.

### Creating a command synonym table

When a user starts a QMF session, QMF loads a command synonym table whose name you specify in the SYNONYMS field of the user's profile.

#### About this task

When you enter a command, QMF first checks the synonyms table for a match. If there is no match, QMF interprets the command as a base QMF command. To prevent QMF from checking the synonyms table, enter the letters QMF in front of any command.

#### Procedure

1. If necessary, acquire or add a table space in the local Db2 for z/OS subsystem to hold the command synonym table.

If you do not have an available table space, create one for your table with a statement like the one shown here:

```
CREATE TABLESPACE DSQTSSN1
  IN DSQDBCTL
  USING STOGROUP DSQSGSYN
  PRIQTY 100
  SECQTY 20
  LOCKSIZE PAGE
  BUFFERPOOL BP0
  CLOSE NO
```

Figure 44. Creating a table space

---

This statement creates the table space DSQTSSN1. The storage group and database for this table space are also those for the table space containing the default synonyms table, Q.COMMAND\_SYNONYMS.

If you use implicit table spaces, skip this step.

2. From the QMF **SQL Query** panel, run a CREATE TABLE statement similar to the following statement. Substitute your own table name in place of COMMAND\_SYNONYMS and your own table space name for DSQTSSN1. If you use implicit table spaces, do not include the IN DSQTSSN1 clause in your statement. Type the other portions of the statement exactly as shown.

```
CREATE TABLE COMMAND_SYNONYMS
( VERB          CHAR(18)      NOT NULL,
  OBJECT        VARCHAR(31),
  SYNONYM_DEFINITION VARCHAR(254) NOT NULL,
  REMARKS       VARCHAR(254) )
IN DSQTSSN1
```

Figure 45. Creating a command synonym table

---

The VERB and OBJECT columns store your synonym. The SYNONYM\_DEFINITION column stores the command or procedure that runs when you enter the synonym.



The columns can be in any order, and you can add a column for comments so that users know the purpose of each synonym.

3. Add comments to the Db2 system catalog using the following example for the COMMAND\_SYNONYMS table:

```
COMMENT ON TABLE COMMAND_SYNONYMS IS 'SYNONYMS FOR R AND D'
```

The phrase SYNONYMS FOR R AND D appears in the REMARKS column of the Db2 system catalog.

You do not need to add comments about your new table to the Db2 system catalog. However, consider adding one comment about the table and other comments to describe the columns. For example, suppose that COMMAND\_SYNONYMS has a column named AUTHID that distinguishes private from public synonyms. To add a comment to explain this, run a query that contains a statement like the following:

```
COMMENT ON COLUMN COMMAND_SYNONYMS.AUTHID  
IS 'PRIVATE SYNONYM: USE AUTH ID. PUBLIC SYNONYM: USE NULL'
```

By issuing a subsequent COMMENT ON statement, you can replace the current comment.

4. Create an index to maximize performance at initialization time, when QMF processes the command synonym table.

Use a statement similar to the following:

```
CREATE UNIQUE INDEX SYNONYMS_INDEX  
ON COMMAND_SYNONYMS (VERB, OBJECT)
```

Index both the VERB and OBJECT columns with the UNIQUE keyword to prevent duplicate synonym definitions. If you choose not to use the UNIQUE keyword, QMF allows duplicate synonyms in the table. QMF uses the first synonym it locates in the table and logs a warning message in the QMF trace data for any duplicates encountered.

## Entering command synonym definitions into the table

After you create a command synonym table, use an SQL INSERT statement to enter your synonyms into the table. You can also use the Table Editor to update the table.

This SQL INSERT statement enters synonyms into the table.

---

```
INSERT INTO COMMAND_SYNONYMS (VERB,OBJECT,SYNONYM_DEFINITION)  
VALUES('COMPUTE','MONTHLY_SALES','RUN PROC JONES.SALES_FIGURES')
```

*Figure 46. Creating a command synonym definition*

---

After it is activated, the synonym COMPUTE MONTHLY\_SALES runs a QMF linear procedure called SALES\_FIGURES, owned by user JONES.

The statement in the following example shows an example of a synonym that has no entry in the object column:

---

```
INSERT INTO COMMAND_SYNONYMS (VERB,SYNONYM_DEFINITION)  
VALUES('EXECUTE','RUN QUERY')
```

*Figure 47. Creating a command synonym definition*

---

After it is activated, the synonym EXECUTE runs the query currently in the QMF temporary storage area.

The synonyms in these examples follow guidelines for synonyms that allow QMF to process each synonym correctly.

## Related concepts

Guidelines for [synonyms](#)

Command synonyms include a verb, a definition, and optionally, an object name.

## Activating the synonyms

Command synonyms follow the same rules for abbreviation as QMF commands.

### About this task

Any abbreviation must indicate a unique QMF command or command synonym. For example, the minimum valid abbreviation for the synonym EXECUTE is EXE. If you enter only EX, QMF cannot distinguish the command synonym EXECUTE from the base QMF command EXPORT.

### Procedure

To activate the command synonym table for your users:

1. Update the SYNONYMS field of the user's profile with the proper command synonym table name.

This column is defined as VARCHAR(261) to allow for a 128-byte table owner ID and table name.

**Important:** Always specify a value for TRANSLATION when you are updating Q.PROFILES, or you might change more rows than you intend.

The following example shows how to assign the COMMAND\_SYNONYMS table to the user JONES in the English language and the table GUMMOW.XYZ to the user SCHMIDT in the German NLF environment:

| Base QMF (English)                                                                                                                 | German NLF                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <pre>UPDATE Q.PROFILES SET SYNONYMS='COMMAND_SYNONYMS' WHERE CREATOR='JONES' AND TRANSLATION='ENGLISH' AND ENVIRONMENT='TSO'</pre> | <pre>UPDATE Q.PROFILES SET SYNONYMS='GUMMOW.XYZ' WHERE CREATOR='SCHMIDT' AND TRANSLATION='DEUTSCH' AND ENVIRONMENT='TSO'</pre> |

The previous statement applies to users who are already enrolled in QMF. You can use a similar statement to update the SYSTEM profile. If you are enrolling a new user, use an INSERT statement.

2. Grant the SQL SELECT privilege to PUBLIC so that assigned users can access the synonyms.

For example:

```
GRANT SELECT ON COMMAND_SYNONYMS TO PUBLIC
```

If you are using a view of a synonyms table rather than the table itself, grant SELECT on only the view to prevent users from accessing synonyms not meant for their use.

### What to do next

Instruct users to end the current QMF session and start another to activate the new synonyms.

### Related tasks

[Assigning views of a synonyms table to individual users](#)

To enable users to have synonyms unique to their needs and still keep table maintenance at an acceptable level, consider creating several views of one synonyms table, and assigning the views to individual users or groups of users. There are three types of views you can create.

## Minimizing maintenance of command synonym tables

---

The command synonym table is initialized before the QMF home panel is displayed. If you notice that QMF initialization time is increasing, you might need to reorganize the command synonym table.

To minimize the time you spend maintaining users' command synonym tables, consider either assigning one synonyms table to all users or assigning a variety of different views of the same table.

### Assigning one synonyms table to all users

The more command synonym tables you create for individual users, the more time you spend on maintenance. One way to reduce maintenance is to create a single command synonym table and assign it to every user.

This statement assigns to every user of base (English)QMF a table with the default name, Q.COMMAND\_SYNONYMS.

```
UPDATE Q.PROFILES
SET SYNONYMS='Q.COMMAND_SYNONYMS'
WHERE TRANSLATION='ENGLISH' and ENVIRONMENT='TSO'
```

Figure 48. Assigning a single command synonym table to all QMF users

---

### Assigning views of a synonyms table to individual users

To enable users to have synonyms unique to their needs and still keep table maintenance at an acceptable level, consider creating several views of one synonyms table, and assigning the views to individual users or groups of users. There are three types of views you can create.

#### Synonyms for public or private use

If you have few synonyms that are used by individuals, consider creating and assigning a view that flags each synonym for either public use (by all users) or private use (by individual users).

#### Procedure

To create and assign a view that flags each synonym for public or private use, follow this procedure:

1. Add an AUTHID column to the synonyms table when you create the table. A null value in the AUTHID column indicates a public synonym; a user ID in the AUTHID column indicates a private synonym. You can have many entries for the same synonym, each assigned to a different user.
2. Use a statement similar to the following one to create a view on the synonyms table. This statement allows a user (indicated by *userid* in the figure) to use all public synonyms in the table and any synonyms assigned privately to his or her SQL authorization ID.

```
CREATE VIEW SYNVIEW (VERB,OBJECT,SYNONYM_DEFINITION)
AS SELECT VERB, OBJECT, SYNONYM_DEFINITION
FROM COMMAND_SYNONYMS
WHERE AUTHID='userid' OR AUTHID IS NULL
```

Figure 49. Creating a view that controls individual and public use of synonyms

---

## Synonyms for public or group use

If you support a large group of end users, consider creating and assigning a view that flags certain synonyms to be used by certain groups of users.

### About this task

The synonyms table that is used to create the view contains a single row for each synonym that belongs to a user group, and a single row for each public synonym. AUTHID is either null or has a value that uniquely identifies the user group.

### Procedure

1. Add an AUTHID column to the synonyms table if it does not have one.
2. Use a statement similar to the following one to create the view on the synonyms table.  
This statement shows a view created for a group of users that have a common user ID, DEPTD02. All users in the DEPTD02 group can use all public synonyms in the table and any synonyms assigned specifically to the group.

```
CREATE VIEW GROUPVIEW (VERB,OBJECT,SYNONYM_DEFINITION)
AS SELECT VERB, OBJECT, SYNONYM_DEFINITION
FROM COMMAND_SYNONYMS
WHERE AUTHID='DEPTD02' OR AUTHID IS NULL
```

*Figure 50. Creating a view that controls group and public use of synonyms*

### Synonyms paired with an authorization table

Consider creating a separate table that holds in one column SQL authorization IDs and the other column the values of a key.

If the keyed value for a particular SQL authorization ID matches a keyed value in a row of the command synonym table, the synonym described in that row is available to the user.

Use a statement similar to the following one to implement this method of maintaining command synonyms. This statement creates a view called KEYVIEW on the table COMMAND\_SYNONYMS, incorporating in the view only the synonyms that have keyed matches between COMMAND\_SYNONYMS and the auxiliary table, KEYTABLE.

```
CREATE VIEW KEYVIEW (VERB,OBJECT,SYNONYM_DEFINITION)
AS SELECT VERB, OBJECT, SYNONYM_DEFINITION
FROM COMMAND_SYNONYMS
WHERE AUTHID IS NULL OR AUTHID IN
(SELECT KEYS FROM KEYTABLE WHERE USER=userid)
```

*Figure 51. Creating a view that uses an extra table to control use of synonyms*

---

## Chapter 15. Custom function keys

The default settings and labels for function keys on each QMF panel describe a common set of QMF tasks that end users are likely to perform. However, because every site's needs are unique, QMF provides a way for you to customize both the label that displays on the screen and the command QMF runs when a user presses the key.

---

### Customizing QMF function keys

You customize the command that QMF issues when a user presses a function key and the key label in the function key table.

#### About this task

QMF function keys are displayed on two types of panels: primary panels, which are full-screen panels such as FORM.MAIN and REPORT, and secondary panels, which open as window dialog panels. Help, prompt, and Prompted Query panels are examples of secondary panels.

### Displaying the panel ID

You enter the panel identifier of the panel for which you want to customize the function keys in the PANEL column of the function key table. Use the global variable DSQDC\_SHOW\_PANID to display the panel ID.

#### About this task

The panel ID is displayed in the upper left corner of the panel when the global variable DSQDC\_SHOW\_PANID is set to 1.

#### Procedure

- Use the following command to set the global variable to display the panel ID:

```
SET GLOBAL (DSQDC_SHOW_PANID=1
```

#### Full-screen panel identifiers

Full-screen panel IDs for the QMF English base product.

Valid full-screen panel IDs for each QMF National Language Feature (NLF) are listed in the language-specific versions of the DSQ22957 message. To view the message for an NLF, issue the command HELP DSQ22957 from the QMF command line while you are running the NLF.

When you enter your function key definitions into the function key table, enter the identifiers exactly as they are shown here or in the message text.

- FORM.BREAK1
- FORM.BREAK2
- FORM.BREAK3
- FORM.BREAK4
- FORM.BREAK5
- FORM.BREAK6
- FORM.CALC
- FORM.COLUMNS
- FORM.CONDITIONS
- FORM.DETAIL
- FORM.FINAL

- FORM.MAIN
- FORM.OPTIONS
- FORM.PAGE
- GLOBALS
- HOME
- PROC
- PROFILE
- PROMPTED QUERY
- REPORT
- SQL QUERY

### Window panel identifiers

Window panel IDs to use when you enter your function key definitions into the function key table.

If you set the global variable DSQDC\_SHOW\_PANID to display the panel IDs, you notice that each ID shown in these tables is prefaced by 4 characters when it is displayed.

Window panels that are not named in the tables do not have unique panel IDs, and can be customized by using the class ID shown at the bottom of each table. All class IDs have the character string XXXX in them. These characters are not variable characters; they are part of the ID.

Choose one of the following sections for more information:

- [“Command windows” on page 240](#)
- [“Forms windows” on page 240](#)
- [“Global variable windows” on page 240](#)
- [“Help and prompt windows” on page 241](#)
- [“Location windows” on page 241](#)
- [“Object list windows” on page 241](#)
- [“Prompted query windows” on page 241](#)

### Command windows

| Panel identifier | Title or description |
|------------------|----------------------|
| COENTR           | Command entry        |
| COXXXX           | Command window class |

### Forms windows

| Panel identifier | Title or description |
|------------------|----------------------|
| FOALIG           | Alignment            |
| FODFIN           | Definition           |
| FOSPEC           | Specify              |
| FOXXXX           | Form window class    |

### Global variable windows

| Panel identifier | Title or description |
|------------------|----------------------|
| GLADVA           | Add variables        |

| Panel identifier | Title or description                                                                                                                                            |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GLSHVA           | Show individual global variables<br>(This label is associated with the panel that displays when a user presses the Show Field key on the <b>GLOBALS</b> panel.) |
| GLXXXX           | Global variables window class                                                                                                                                   |

### Help and prompt windows

| Panel identifier | Title or description                                                                           |
|------------------|------------------------------------------------------------------------------------------------|
| HEXXXX           | Subject help window class<br>(Error message help panels follow a different naming convention.) |
| PRXXXX           | Prompt window class                                                                            |

### Location windows

| Panel identifier | Title or description |
|------------------|----------------------|
| PLLOCA           | Location window list |

### Object list windows

| Panel identifier | Title or description         |
|------------------|------------------------------|
| OBDESC           | Object description           |
| OBLIAC           | Object list action           |
| OBLIMU           | Object list multi-selection  |
| OBLISI           | Object list single-selection |
| OBSORT           | Object list sort             |
| OBXXXX           | Object list window class     |

### Prompted query windows

| Panel identifier | Title or description          |
|------------------|-------------------------------|
| QPCDCH           | Condition connector - change  |
| QPCDIT           | Condition connector           |
| QPCFOR           | Change time period            |
| QPCOCH           | Column - change               |
| QPCODE           | Column description            |
| QPCOFI           | Column summary function items |
| QPCOFU           | Column summary functions      |
| QPCOLI           | Column names list             |
| QPCOLU           | Columns                       |
| QPDUCH           | Duplicate rows - change       |
| QPDUPL           | Duplicate rows                |

| Panel identifier | Title or description                   |
|------------------|----------------------------------------|
| QPERSP           | Specify time period                    |
| QPEXPR           | Expression                             |
| QPJOCO           | Join columns                           |
| QPJOTA           | Join tables                            |
| QPROBE           | Rows - between                         |
| QPROCH           | Rows - change (left side)              |
| QPROCT           | Rows - containing                      |
| QPROC1           | Rows - comparison operators 1          |
| QPROC2           | Rows - comparison operators 2          |
| QPROEN           | Rows - Ending with                     |
| QPROEQ           | Rows - Equal to                        |
| QPROGQ           | Rows - Greater than or equal to        |
| QPROGR           | Rows - Greater than                    |
| QPROLQ           | Rows -Less than or equal to            |
| QPROLS           | Rows - Less than                       |
| QPROST           | Rows - Starting with                   |
| QPROWS           | Rows (row conditions)                  |
| QPSHFI           | Show field                             |
| QPSHSQ           | Show SQL                               |
| QPSOCH           | Sort - change                          |
| QPSORT           | Sort                                   |
| QPSPEC           | Specify                                |
| QPTABL           | Tables                                 |
| QPTTAB           | Select tables or views for time period |
| QPXXXX           | PQ window class                        |

## Choosing the keys that you want to customize

The default key labels differ by panel type. Choose the function key that you want to customize.

You cannot customize function keys on Table Editor panels. On other panels, you can choose QMF or site-defined commands to associate with any function key label you modify.

### Default keys on full-screen panels

This table shows the default function key settings for full-screen panels.

| Default key label | Default command issued when key is pressed |
|-------------------|--------------------------------------------|
| Backward          | BACKWARD                                   |
| Cancel            | CANCEL                                     |



Table 53. Default function keys and commands for full-screen panels (continued)

| <b>Default key label</b> | <b>Default command issued when key is pressed</b> |
|--------------------------|---------------------------------------------------|
| Change                   | CHANGE                                            |
| Chart                    | DISPLAY CHART or SHOW CHART                       |
| Check                    | CHECK                                             |
| Clear                    | CLEAR                                             |
| Command                  | SHOW COMMAND                                      |
| Comments                 | SWITCH COMMENTS                                   |
| Delete                   | DELETE                                            |
| Describe                 | DESCRIBE                                          |
| Draw                     | DRAW                                              |
| Edit Table               | EDIT TABLE                                        |
| End                      | END                                               |
| Enlarge                  | ENLARGE                                           |
| Form                     | DISPLAY FORM or SHOW FORM                         |
| Forward                  | FORWARD                                           |
| Help                     | HELP                                              |
| Insert                   | INSERT                                            |
| Left                     | LEFT                                              |
| List                     | LIST                                              |
| Print                    | PRINT                                             |
| Proc                     | DISPLAY PROC or SHOW PROC                         |
| Profile                  | DISPLAY PROFILE                                   |
| Query                    | DISPLAY QUERY or SHOW QUERY                       |
| Reduce                   | REDUCE                                            |
| Refresh                  | REFRESH                                           |
| Report                   | DISPLAY REPORT or SHOW REPORT                     |
| Retrieve                 | RETRIEVE                                          |
| Right                    | RIGHT                                             |
| Run                      | RUN QUERY or RUN PROC                             |
| Save                     | SAVE PROFILE                                      |
| Show                     | SHOW                                              |
| Show Field               | SHOW FIELD                                        |
| Show SQL                 | SHOW SQL                                          |
| Sort                     | SORT                                              |
| Specify                  | SPECIFY                                           |

Table 53. Default function keys and commands for full-screen panels (continued)

| Default key label | Default command issued when key is pressed |
|-------------------|--------------------------------------------|
| Specify View      | SPECIFY VIEW                               |

### Default keys on window panels

This table shows the default function keys on window panels as well as the command executed when each key is pressed.

Table 54. Default function keys and commands for window panels

| Default key label   | Default command issued when key is pressed |
|---------------------|--------------------------------------------|
| Attribute           | SPECIFY ATTRIBUTES                         |
| Backward            | BACKWARD                                   |
| Cancel              | CANCEL                                     |
| Clear               | CLEAR                                      |
| Command             | SHOW COMMAND                               |
| Comments            | SWITCH COMMENTS                            |
| Condition           | SPECIFY CONDITION                          |
| Delete              | DELETE                                     |
| Describe            | DESCRIBE                                   |
| End                 | END                                        |
| Exit                | END                                        |
| Forward             | FORWARD                                    |
| Help                | HELP                                       |
| Index               | HELP INDEX                                 |
| Keys                | HELP KEYS                                  |
| List                | LIST                                       |
| Menu                | HELP MENU                                  |
| More Help           | MORE HELP                                  |
| Next Column         | NEXT COLUMN                                |
| Next Definition     | NEXT DEFINITION                            |
| Previous Column     | PREVIOUS COLUMN                            |
| Previous Definition | PREVIOUS DEFINITION                        |
| Refresh             | REFRESH                                    |
| Show Entity         | SHOW ENTITY                                |
| Show Field          | SHOW FIELD                                 |
| Show View           | SHOW VIEW                                  |
| Sort                | SORT                                       |
| Specify Attributes  | SPECIFY ATTRIBUTES                         |

Table 54. Default function keys and commands for window panels (continued)

| Default key label | Default command issued when key is pressed |
|-------------------|--------------------------------------------|
| Specify Condition | SPECIFY CONDITION                          |
| Switch            | HELP SWITCH                                |

On the global variable list panel, RESET GLOBAL is the command executed when the Delete function key is pressed.

## Creating the function key table

After you decide which function keys you want to customize, you can create a table that links your customized function key definitions with the appropriate panels.

### Procedure

Follow these steps to create the function key table:

1. Use an SQL CREATE TABLE statement similar to the one shown here to create the table. Substitute your own name for MY\_PFKEYS. Substitute your own table space for DSQTSSN1.

```
CREATE TABLE MY_PFKEYS
(PANEL          CHAR(18)          NOT NULL,
 ENTRY_TYPE    CHAR(1)          NOT NULL,
 NUMBER        SMALLINT         NOT NULL,
 PF_SETTING    VARCHAR(254),
 REMARKS       VARCHAR(254))
IN DSQTSSN1
```

Figure 52. Creating a function key table

2. Add comments to the Db2 system catalog using an SQL statement similar to the following statement:

```
COMMENT ON TABLE MY_PFKEYS IS 'PF KEYS RESERVED FOR FINANCIAL ANALYSTS'
```

The phrase PF KEYS RESERVED FOR FINANCIAL ANALYSTS appears in the REMARKS column of the Db2 system catalog.

You do not need to add comments about your new table to the Db2 system catalog; however, if you do, one comment might be about the table and others might describe the columns. For example, suppose that MY\_PFKEYS has a column named AUTHID that distinguishes private from public function keys. To add a comment to explain this, run a query containing a statement like the one in the following example:

```
COMMENT ON COLUMN MY_PFKEYS.AUTHID
IS 'PRIVATE PFKEY: USE AUTH ID. PUBLIC PFKEY: USE NULL'
```

By issuing a subsequent COMMENT ON statement, you can replace the current comment.

3. Create an index using an SQL statement similar to the following statement:

```
CREATE UNIQUE INDEX MY_PFKEYSX
ON MY_PFKEYS (PANEL, ENTRY_TYPE, NUMBER)
```

Use the UNIQUE keyword to index the PANEL, ENTRY\_TYPE, and NUMBER columns to ensure that no two rows of the table can be identical.

If you choose not to use the UNIQUE keyword, QMF allows duplicate key definitions. QMF logs warning messages if it finds more than one key definition for the same key, and writes these messages to the user's trace data. Multiple key definitions for window panels cause no messages; QMF uses the last definition it finds.

## Entering your function key definitions into the table

You can use SQL INSERT statements or the QMF Table Editor to insert key definitions into the function key table.

Each function key definition spans two rows in the table:

- One row specifies the command QMF issues when a user presses the key.
- The other row specifies the key label that appears on the screen.

Enter both rows for each key that you want to customize. A function key command without an associated label doesn't appear on the user's screen. Similarly, a label with no associated command is inactive.

### Related concepts

#### Examples of key definitions

Use the examples in this topic to see how to enter a complete function key definition for each type of QMF panel. The examples show how to update a full-screen panel, a window panel, and a help panel.

### Linking a command with a function key

Each function key on a QMF panel is linked with a QMF command that executes when the function key is pressed. To ensure that the function key is active, add a row to the function key table that identifies the command.

To ensure your customized function keys also work this way, make sure that one of the two rows you enter into the table has the values shown in the following table.

| Column     | Value                                       | Information                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PANEL      | ID of the <b>QMF</b> panel to be customized | If you want to define the same set of keys to appear on every panel in a class of window panels, use the class ID shown at the bottom of the tables. For example, to customize the <b>Specify</b> panel of a <b>Forms</b> window, use the panel ID FOSPEC if you want the <b>Specify</b> panel to have different keys than the rest of the panels in the forms class. Otherwise, use the panel ID FOXXXX, which characterizes all panels in that class.<br><br>Changes that you make using a class ID apply to all panels customized by that class ID. Help and prompt windows do not have a set of unique IDs; they can be customized only by using class IDs. |
| ENTRY_TYPE | K                                           | K indicates that this row defines the command that QMF issues when the key is pressed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| NUMBER     | Number of the function key to be customized | For example, if you are changing the definition for F5, enter a 5 in this column.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

Table 55. Values for the first of two rows required for each function key definition (continued)

| Column     | Value                                                 | Information                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------|-------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PF_SETTING | Text of the command that runs when the key is pressed | <p>Make sure that this command is appropriate for the panel on which it appears. For example, the ENLARGE command is appropriate only for the <b>QUERY</b> panel in a QBE query. Because QMF does not check if the command is appropriate for the panel until the user presses the key, test each of your new function keys before their first use.</p> <p>Enter the command in upper case, because QMF does not convert display device input to upper case when it retrieves the commands associated with function keys. The command will not run if this value is in lower case and the CASE field of the user's profile has the value UPPER.</p> <p>Ensure that each panel that you customize has a key set to END or CANCEL. Without a key defined to one of these commands, users might not be able to exit the panel.</p> |

If you are using an NLF, make sure that the underlying command has the correct national language translation and that the label text for each key is written in the language of the NLF you are using.

**Related reference**

Labeling the function key and positioning it on the screen

The function keys on each QMF panel have labels next to the function key numbers. To ensure that the label appears on the screen, add a row to the function key table that identifies the label.

**Labeling the function key and positioning it on the screen**

The function keys on each QMF panel have labels next to the function key numbers. To ensure that the label appears on the screen, add a row to the function key table that identifies the label.

In this row, make sure that the columns of the function key table have the values shown in the following table.

Table 56. Values for the second of two rows required for each function key definition

| Column     | Value                                                                                | Information                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PANEL      | ID of the QMF panel to be customized                                                 | This is the same ID that you used for the first row of the definition.                                                                                                                                                                                                                                                                                                                                                           |
| ENTRY_TYPE | L                                                                                    | L indicates that the row defines the label associated with the function key.                                                                                                                                                                                                                                                                                                                                                     |
| NUMBER     | Depends on whether you are customizing a full-screen panel or a window or help panel | <p>If you are customizing a full-screen panel, this is the number of the row where the key appears on the display. For example, on the home panel, F5 appears in row 1, so the NUMBER column in this definition would have a value of 1.</p> <p>If you are customizing a window or help panel, NUMBER represents the number of the function key (as it does in the first row you added to the table that links the command).</p> |

Table 56. Values for the second of two rows required for each function key definition (continued)

| Column     | Value                           | Information                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PF_SETTING | Text of the function key labels | <p>For full-screen panels, QMF displays on the screen exactly what you enter in this column, and does not adjust for spacing. For example, if you are customizing the QMF home panel, you need to enter all the keys that appear on that panel, whether or not you customized them. QMF does not automatically assign the default key settings for keys that you do not customize. See <a href="#">Figure 53 on page 249</a> for an example.</p> <p>For window panels, you need to type only the label of the key in this column. See <a href="#">Figure 56 on page 250</a> and <a href="#">Figure 57 on page 250</a> for examples.</p> |

### Related reference

#### Linking a command with a function key

Each function key on a QMF panel is linked with a QMF command that executes when the function key is pressed. To ensure that the function key is active, add a row to the function key table that identifies the command.

## Activating new function key definitions

To enable users to use the customized function key definitions that you created, you must activate them.

### Procedure

1. Update the PFKEYS field of the user's profile with the name of your function key definitions table.

In the following example, the statement on the left assigns to English QMF user JONES the table MY\_PFKEYS, and the statement on the right assigns to German NLF user SCHMIDT the table MEIN\_PFKY.

**Important:** Always include a value for the TRANSLATION and ENVIRONMENT columns in a statement that updates the Q.PROFILES table or you might change more rows than you intend.

| Table 57. Making customized function keys accessible to a user                                                                  |                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Base QMF (English)                                                                                                              | German NLF                                                                                                                        |
| <pre>UPDATE Q.PROFILES SET PFKEYS = 'MY_PFKEYS' WHERE CREATOR='JONES' AND TRANSLATION = 'ENGLISH' AND ENVIRONMENT = 'TSO'</pre> | <pre>UPDATE Q.PROFILES SET PFKEYS = 'MEIN_PFKY' WHERE CREATOR='SCHMIDT' AND TRANSLATION = 'DEUTSCH' AND ENVIRONMENT = 'TSO'</pre> |

2. Grant the SQL SELECT privilege to users who need to access the table.

To allow any user to whom the table is assigned to use it, grant the SELECT privilege to PUBLIC. For example:

```
GRANT SELECT ON MY_PFKEYS TO PUBLIC
```

To minimize maintenance of function keys at your site, you can assign a view of the table. Grant the SELECT privilege on only the view to prevent users from accessing function keys not meant for their use.

Use the strategies for assigning views of a function key table to decide whether to assign a table or a view to individual users or groups of users.

### What to do next

Instruct users to end the current QMF session and start another to activate the new function keys.

## Related tasks

Assigning views of a synonyms table to individual users

To enable users to have synonyms unique to their needs and still keep table maintenance at an acceptable level, consider creating several views of one synonyms table, and assigning the views to individual users or groups of users. There are three types of views you can create.

## Testing and diagnosing problems with the function key table

After you have activated the new function key definitions by inserting the function key table name into the user's Q.PROFILES entry, the new definitions are ready to be tested.

### Procedure

You can test the new definitions one of two ways:

- Exit QMF and start a new QMF session.
- From within QMF, reconnect by entering the `CONNECT TO servername` command, where *servername* is the same location name that you see on the QMF home panel.

### Results

If you do not see the new function key definitions after reconnecting to QMF, review the QMF trace output for possible error or warning messages.

If the QMF trace data shows no errors, issue the `SHOW GLOBALS` command and check the global variable `DSQAP_PFKEY_TABLE`. If this global variable does not contain the name of the newly created or modified function key table, review the user's row in the `Q.PROFILES` table.

## Examples of key definitions

---

Use the examples in this topic to see how to enter a complete function key definition for each type of QMF panel. The examples show how to update a full-screen panel, a window panel, and a help panel.

The examples shown use panel IDs from the tables that list the full screen panels and window panels.

### Entering a definition for a key on a full-screen panel

Use the following SQL statements to change F2 on the home panel from `EDIT TABLE` to `IMPORT`. Identify the home panel with the panel ID `HOME`, and indicate with the number 2 (in the first statement shown) that you want to customize the command executed when a user presses F2.

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('HOME','K',2,'IMPORT')
```

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('HOME','L',1,'1=Help 2=Import 3=End 4=Show 5=Chart 6=Query')
```

Figure 53. Changing a function key for a QMF command on the home panel

---

The QMF home panel now displays `Import` for F2, as shown here:

Type command on command line or use PF keys. For help, press PF1 or type HELP.

```
-----  
1=Help      2=Import    3=End       4=Show      5=Chart     6=Query  
7=Retrieve  8=Edit Table 9=Form      10=Proc     11=Profile  12=Report  
OK, cursor positioned.  
COMMAND ===>
```

Figure 54. After adding the definition to the function keys table, the function key appears customized on the screen

In the PF\_SETTING column of the second INSERT statement, be sure to type exactly what appears in the top row of keys on the home panel, even if you have not customized each key. For example, if you specify only the word **Import** in the PF\_SETTING column for the second statement, the home panel looks like the following one:

Type command on command line or use PF keys. For help, press PF1 or type HELP.

```
-----  
Import  
7=Retrieve  8=Edit Table 9=Form      10=Proc     11=Profile  12=Report  
OK, cursor positioned.  
COMMAND ===>
```

Figure 55. The first row of keys is missing if you do not specify them when you insert the new key definition into the function keys table

### Entering a definition for a key on a window panel

The following SQL statements add an F3 key to the **Tables** panel in Prompted Query. The function key executes the CANCEL command, and is labeled CancelMe.

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)  
VALUES('QPTABL', 'K', 3, 'CANCEL')
```

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)  
VALUES('QPTABL', 'L', 3, 'CancelMe')
```

Figure 56. Changing a function key on the **Specify** panel for Prompted Query

### Entering a key definition for a help or prompt panel

The following SQL statements in add an F13 key to all help panels. The function key executes the CANCEL command, and is labeled CancelMe.

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)  
VALUES('HEXXXX', 'K', 13, 'CANCEL')
```

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)  
VALUES('HEXXXX', 'L', 13, 'CancelMe')
```

Figure 57. Changing a function key on a help panel or prompt panel

All help and prompt panels are customized using a single class ID. Because any changes you make to one panel in the class appear on all panels that are defined with that class ID, ensure that changes you make to one help or prompt panel are appropriate for all the help and prompt panels in that class.

### Related reference

[Full-screen panel identifiers](#)

Full-screen panel IDs for the QMF English base product.

[Window panel identifiers](#)



Window panel IDs to use when you enter your function key definitions into the function key table.



# Chapter 16. Custom edit exit routines for QMF forms

QMF forms help users to control the format of data returned from the database. If the default edit codes do not meet the report-editing needs of your site, you can create your own edit codes.

You can use edit codes in the EDIT field of the FORM.MAIN and FORM.COLUMNS panels to format report data in different ways. For example, you can use a decimal edit code for a column that returns salary data; this edit code formats the numeric data into a decimal with a currency symbol.

## Edit exit routines and QMF

QMF and your edit exit routine work together to format data by using the edit codes that you define.

### Calling your exit routine to format the data

This image shows how a user edit routine works with QMF.

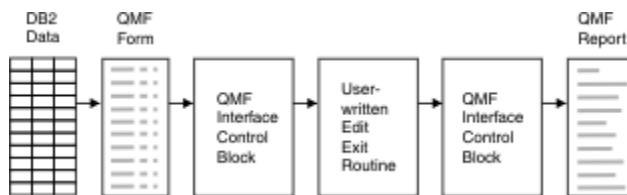


Figure 58. How a user edit routine works with QMF for TSO and CICS

When you enter your own code in a column of FORM.MAIN or FORM.COLUMNS, QMF passes certain characteristics of the data into the first interface control block. These characteristics reside in specific fields of the control block. QMF also passes into the input area the data to be formatted and an output area that holds the formatted result.

### Passing information to and from the exit routine

To format the data returned from the database, QMF calls your edit exit routine and passes information through fields of the interface control block. Information is also passed to and from the exit routine using the input and output areas, which contain the data to be formatted and information about where to put the formatted result.

The data to be formatted can be a column value, the result of a built-in function, a defined column, a calculation, or a value represented by a variable in a heading, a footing, or a final-summary line.

Upon receiving control for formatting, your edit routine receives the following parameters:

- Control information from the interface control block
- The value of ECSINPT, the data from the input area to be formatted
- ECSRSLT, which contains the output area for the formatted result

ECSRLEN contains the length of the output area. If your formatted result doesn't fit in the output area, it must be truncated because the output area cannot be column-wrapped.

**Important:** Do not use more memory in the output area than is indicated by the value in the ECSRLEN field, or you will see error DSQ60439: User edit program memory overwrite. To correct this error, do one of the following:

- Increase the width of the column by modifying the edit code in the QMF form to the length expected on the report.
- Code your edit exit program so that it checks the value of the ECSRLEN field to determine if your program should pad or truncate the results passed back to QMF.

ECSINPT, ECSRSLT, and ECSRSLN are fields of the interface control block.

### Passing control to the exit routine when QMF terminates

Use the ECSRQMF field of the control block to indicate that you want your exit routine to receive control when QMF terminates.

The ECSRQMF value should be updated the first time the edit exit routine receives control.

When your edit exit routine receives control upon termination of QMF, the parameters passed to the routine are the control block, the input area, and the output area. Only the control block contains usable information.

### Related concepts

[Fields that characterize the input and output areas](#)

In addition to the fields in the interface control block, your edit exit routine receives, in the input field, information about the data to be formatted. The output field contains information about where to put the formatted result.

### Related reference

[Fields of the interface control block](#)

Use the fields of the interface control block to pass information to and from your exit routine.

## Fields passed to and from the exit routine

---

These fields of the interface control block and of the input and output areas contain information that is passed either to or from your edit exit routine.

### Fields of the interface control block

Use the fields of the interface control block to pass information to and from your exit routine.

Although there are separate interface control blocks that work with Assembler, PL/I, or COBOL, the fields of the interface control block are standard regardless of the programming language in which your edit exit routine is written. These fields are shown in the following table. Unless otherwise stated, each field relates to all formatting calls.

Language-specific versions of the edit exit interface control structure are located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language):

- For High-Level Assembler, use DSECT DXEECSA
- For PL/I, use copy file DXEECS
- For COBOL, use copy book DXEECS

| Name            | Description                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------------------|
| <b>ECSDECPT</b> | Contains the current decimal point symbol as determined by the DECOPT option of the QMF profile (period or comma). |
| <b>ECSECODE</b> | Contains the user edit code.                                                                                       |

Table 58. Fields of the QMF edit exit interface control block, DXEECS (continued)

| Name            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ECSERRET</b> | <p>Contains a zero at the point of call. Set this to a nonzero return code to record an error. Use one of the values in the following list for an error of the indicated type:</p> <p><b>Number</b><br/><b>Error</b></p> <p><b>99101</b><br/>Unrecognized edit code</p> <p><b>99102</b><br/>Improper input data type for edit code</p> <p><b>99103</b><br/>Invalid input value for item to be formatted</p> <p><b>99104</b><br/>Item to be formatted is too short</p> <p><b>99105</b><br/>Not enough room for result in ECSRSLT (result is too wide for the space allotted)</p> <p>The error codes listed (and their associated messages and help panels) are specific to the error. For any other code, a general error message, with a general help panel, is displayed.</p> |
| <b>ECSFREQ</b>  | Holds E for a formatting call, T for a termination call.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>ECSINDTA</b> | Contains information about the value to be formatted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>ECSINLEN</b> | Contains the length, in bytes, of the value to be formatted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>ECSINNUL</b> | Holds an N if the value to be formatted is null.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>ECSINPRC</b> | Contains the precision of the value to be formatted. Applies only to U-type codes when the data type is DECIMAL, or to V-type codes when the character string to be formatted was derived from numeric data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>ECSINSCL</b> | Contains the scale of the value to be formatted. Applies only to U-type codes when the data type is DECIMAL, or to V-type codes when the character string to be formatted was derived from numeric data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>ECSINSGN</b> | Holds the sign of a converted numeric value (blank or - ). Applies only to V codes when the character string to be formatted was derived from numeric data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

Table 58. Fields of the QMF edit exit interface control block, DXEECS (continued)

| Name            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ECSINTYP</b> | <p>The internal database representation of the data type of the value to be formatted. The code shown here for each data type is the same as the value that appears in the SQLTYPE field of the SQLDA.</p> <p><b>384</b><br/>DATE data type</p> <p><b>388</b><br/>TIME data type</p> <p><b>392</b><br/>TIMESTAMP data type</p> <p><b>448</b><br/>VARCHAR data type</p> <p><b>452</b><br/>CHAR data type</p> <p><b>456</b><br/>LONG VARCHAR data type</p> <p><b>464</b><br/>VARGRAPHIC data type</p> <p><b>468</b><br/>GRAPHIC data type</p> <p><b>472</b><br/>LONG VARGRAPHIC data type</p> <p><b>480</b><br/>FLOAT data type</p> <p><b>484</b><br/>DECIMAL data type</p> <p><b>492</b><br/>BIGINT data type</p> <p><b>496</b><br/>INTEGER data type</p> <p><b>500</b><br/>SMALLINT data type</p> <p><b>908</b><br/>VARBINARY data type</p> <p><b>912</b><br/>BINARY data type</p> <p><b>940</b><br/>Extended FLOAT</p> <p><b>996</b><br/>DECFLOAT data type (both long format and extended format)</p> <p><b>2448</b><br/>TIMESTAMP WITH TIME ZONE data type</p> |
| <b>ECSNAME</b>  | Contains the name of the control block, which is DXEECS. Serves as an eye catcher in storage dumps.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>ECSRQMF</b>  | Set this to T to request a termination call.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>ECSRSDTA</b> | Contains information about the formatted result.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

Table 58. Fields of the QMF edit exit interface control block, DXEECS (continued)

| Name            | Description                                                                                                                                                                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ECSRSLEN</b> | Contains the length of the output area, in bytes. (This value is taken from the WIDTH column of the QMF form.) Do not use more memory in the output area than is indicated by the value in the ECSRSLEN field, or you will see error DSQ60439: User edit program memory overwrite. |
| <b>ECSTHSEP</b> | Contains the thousands separator as determined by the DECOPT option of the QMF profile (blank or a comma).                                                                                                                                                                         |
| <b>ECSUSERS</b> | A 256-byte scratchpad area where your exit routine can record information that persists from one call to the next. On the first call after the edit routine is loaded, this field contains binary zeros.                                                                           |

### Related concepts

[Edit exit routines and QMF](#)

QMF and your edit exit routine work together to format data by using the edit codes that you define.

### Related tasks

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Fields that characterize the input and output areas

In addition to the fields in the interface control block, your edit exit routine receives, in the input field, information about the data to be formatted. The output field contains information about where to put the formatted result.

The value to be formatted appears in the ECSINPT field. How it is represented depends on whether the value is numeric or character, as determined by the ECSINTYP field, or whether the edit code is a U or V code, as determined by the ECSECODE field.

### How U-type edit codes are represented in the input area

Numeric values are represented in internal database format. For example, if ECSINTYP is equal to 496 (INTEGER data type), the value is a full-word integer. If it is 484 (DECIMAL data type), the value is in decimal format. Scale and precision for decimal format are in the ECSINSCL and ECSINPRC fields. Length (in bytes) is in the ECSINLEN field.

Numeric data from defined columns, calculations, and summary values is returned as extended floating-point values. The length (16 bytes) is in the ECSINLEN field.

Character or graphic values are represented in their internal, character-string format, with one exception: for variable-length strings (for example, the VARCHAR data type), only the string itself appears and not the preceding length field. For all character values, the string length (in bytes) is in the ECSINLEN field.

### How V-type edit codes are represented in the input area

Numeric values are represented by a numeric character string. The length is contained in the ECSINLEN field. Leading or trailing zeros fill out the string if required.

The string contains no sign or decimal point. Instead, the sign appears as a blank or a minus sign in the ECSINSGN field, and the position of the decimal point is in the ECSINSCL field. For example, suppose that the string in ECSINPT is 12345, that ECSINSGN is blank, and that ECSINSCL is equal to 3; then the value represented is +12.345.

Character or graphic values are represented in their character strings. For all character values, the string length (in bytes) is in the ECSINLEN field.

## The output area

The ECSRSLT field receives the formatted output in the form of a character string that completely fills the field. This result is then returned in a QMF report. The length of this field (in bytes) is stored in the ECSRLEN field. This field is always blank before the edit routine is called.

## Related concepts

[Edit exit routines and QMF](#)

QMF and your edit exit routine work together to format data by using the edit codes that you define.

## Choosing an edit code

---

You can create your own edit codes for data of all types except XML, BLOB, CLOB, and DBCLOB. You can create your own edit codes for decimal floating-point data types only if both the processor on which QMF is running and the language in which you write the exit routine support decimal floating-point instructions.

### About this task

An edit code is a set of characters that tells QMF how to format and punctuate the data in a specific column of a report.

You do not need to restrict an edit code to the processing of numeric data, or to the processing of character data. The sample edit routines supplied with QMF process one edit code for both numeric and character data.

If the CASE field of a user's profile has the value UPPER or STRING, QMF converts all input entered from the display device to upper case. If your edit routine is written to accept edit codes in mixed case, enter the edit codes when CASE is set to MIXED.

### Procedure

- Decide what you want your routine to do and choose either a Uxxxx or Vxxxx edit code to associate with the routine that will format the data.
  - For U codes, data passed to the edit routine has the internal database representation of the source data unless the field on which the user edit code is used is the result of an expression. All data resulting from expressions is passed back to the edit routine as extended floating-point data.
  - For V codes, numeric data is converted to a character string, and this character string is passed to the edit program.

Both codes can be used to process either character or numeric data. U and V must be in upper case.

When the source data is character, codes of either type are equally easy to process. If the formatting requires arithmetic operations, consider using U codes for numeric sources; otherwise, use V codes.

If the programming language you are using does not support a particular data type, use a V edit code to convert those values to characters. For V codes containing numeric data, QMF converts the data to character format and then calls the user edit routine. The length of the converted number varies depending upon its original data type, as shown in this table:

| <b>If data type of original numeric data is:</b> | <b>QMF converts it to this length:</b>                                                             |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Small integer                                    | 6                                                                                                  |
| Integer                                          | 10                                                                                                 |
| Big integer                                      | 19                                                                                                 |
| Decimal                                          | Equal to the precision of the original data (raised to an odd number if the original data is even) |



| <i>Table 59. How QMF converts numeric data according to data type (continued)</i> |                                               |
|-----------------------------------------------------------------------------------|-----------------------------------------------|
| <b>If data type of original numeric data is:</b>                                  | <b>QMF converts it to this length:</b>        |
| Floating point                                                                    | 15 or more, depending on the base 10 exponent |
| Extended floating point                                                           | 30 or more, depending on the base 10 exponent |
| Long-format decimal floating point                                                | 17                                            |
| Extended-format decimal floating point                                            | 31                                            |

- Replace xxxx with zero to four characters (letters, digits, or special characters); embedded blanks or nulls are not allowed.

The following examples show valid U-type and V-type edit codes:

```
U1
UAB42
V_1
VX%5
```

## Double-byte character set data and edit routines

Double-byte character set (DBCS) data can appear in character columns or in columns with a graphic data type (GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC). If you need to devise edit routines that process this type of data, ensure that your routine handles the data correctly.

Among the characters represented by the Japanese DBCS are Latin characters and Katakana characters. A Latin character has these characteristics:

- The first (left-most) byte of the character has the value X'42'.
- The second byte of the character contains the EBCDIC equivalent.

A Katakana character has these characteristics:

- The first byte of the character contains X'43'.
- The second byte contains the EBCDIC equivalent.

You can use either Uxxxx or Vxxxx edit codes for DBCS data. The data that the edit routine receives is the same.

### DBCS data and what the edit routine receives

The data to be formatted is in the field ECSINPT, and the length of that data, in bytes, is in ECSINLEN.

What you find in ECSINPT depends, to some extent, on where the data originates. More precisely, it depends on whether the column containing that data is a character column or one with a graphic data type.

#### Data from character columns

If the data to be processed comes from a character column, then the data in ECSINPT is just a copy of the column data. Unlike data from a graphic column, this data can hold single-byte characters and shift characters, as well as DBCS characters. To locate DBCS characters, you must search for the S0 and SI characters that bracket the DBCS strings. If there are no S0 or SI characters in ECSINPT, the string contains no DBCS data. For example, suppose that ECSINPT contains the following string:

```
ccccSodededdedededeSiccSodededdededeSi
```

Here, c, d, and e stand for any possible byte, and S0 and SI are shift bytes. From the placement of the shift bytes, you can see that every occurrence of c represents a single-byte character, and that every occurrence of de represents a DBCS character.

Single-byte characters can represent Latin letters, Arabic numerals, and special characters such as plus signs and parentheses. For Japanese character sets, they can also be Katakana characters. Some bytes meant to represent lowercase Latin might be displayed as Katakana symbols. You might have to devise edit codes that distinguish between columns containing lowercase English and those containing Katakana.

### **Data from graphic columns**

If the data to be formatted is from a column with a graphic data type, then the text in ECSINPT consists of this data preceded by one shift character and followed by another. Both shift characters are single bytes. For DBCS display devices, shift characters mark the start and end of a string of DBCS characters.

S0 denotes the shift character that introduces a DBCS string, and SI denotes the one that marks its end. S0 has the value X'0E'. SI has the value X'0F'. The shift characters are included in the data length recorded in ECSINLEN.

Thus, the length appearing in ECSINLEN is always greater by two than the length of the actual data. Because the data is presumably a string of DBCS characters, its length (in bytes) is always an even number.

### **Ensuring that the edit routine returns the right results**

You must return the edited results in the ECSRSLT field, with trailing blanks for unused bytes, and ensure that the user's display device has the capability to read them.

This means that the resulting DBCS and EBCDIC characters must have the appropriate representations, and that the beginning and end of any string of DBCS characters are marked by S0 and SI characters.

### **Overflowing the ECSRSLT field**

Be careful not to overflow the ECSRSLT field, whose length is contained in the ECSRLEN field. If your results do not fit, truncate them on the right. If the last character represented in the truncated results is a DBCS character, be certain to retain its right-most byte, and to follow that character with an SI character.

### **Printing the report column**

QMF copies the ECSRSLT field into the corresponding report column. The result is exactly as wide as the report column. If you do not specify ALIGNMENT for data, the data is aligned exactly as you typed it.

How the report device represents what you return depends on the specific device. For many display devices, the following rules apply:

- If the report is displayed on the screen, the SI and S0 characters embedded in a user's results also appear on the display.
- The SI and S0 characters appear either as blanks or as special symbols. There is one special symbol for SI and another for S0.
- Blanks appear instead of the symbols unless the user presses a certain combination of keys.

Any legitimate DBCS character can be returned in the ECSRSLT field.

## **Date, time, and timestamp data and edit routines**

---

If you are writing an edit exit routine to format DATE, TIME, TIMESTAMP, or TIMESTAMP WITH TIME ZONE data, you must use the default formats for each data type. If you do not require formatting of

TIMESTAMP or TIMESTAMP WITH TIME ZONE data, you can also the exit routines supplied with Db2 for date and time data.

## Required formats for date, time, and timestamp information

Your edit routine can format data from datetime columns, just as it can format data from columns with other data types. The one difference is that the value to be formatted, which appears in the control block field ECSINPT, is always passed as a character string, whether the code to be processed is a U code or a V code.

The expected format of the string is described in this table.

| <i>Table 60. Formatting DATE, TIME, and TIMESTAMP data</i> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Data type</b>                                           | <b>Form of the string</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| DATE data                                                  | <p><i>yyyy-mm-dd</i> where:</p> <p><b>yyyy</b><br/>Specifies the year. It is always a four-digit number.</p> <p><b>mm</b><br/>Specifies the month (01 for January, 12 for December). It is always a two-digit number that can contain a leading zero.</p> <p><b>dd</b><br/>Specifies the day of the month. It is always a two-digit number that can contain a leading zero.</p> <p>The dashes (-) represent true dashes.</p> <p>For example, 2006-03-12 is the date March 12, 2006.</p>                                                                           |
| TIME data                                                  | <p><i>hh.mm.ss</i> where:</p> <p><b>hh</b><br/>Specifies the hour (based on a 24-hour clock, from 00 to 23). It is always a two-digit number that can contain a leading zero.</p> <p><b>mm</b><br/>Specifies the minute. It is always a two-digit number that can contain a leading zero.</p> <p><b>ss</b><br/>Specifies the second. It is always a two-digit number that can contain a leading zero.</p> <p>The periods represent true periods.</p> <p>For example, 13.08.36 is 1:08 P.M. and 36 seconds in the notation commonly used in the United States.</p> |
| TIMESTAMP data                                             | <p><i>yyyy-mm-dd-hh.mm.ss.nnnnnnnnnnnn</i> where:</p> <p><b>yyyy-mm-dd</b><br/>Specifies the date in the same way as for DATE data.</p> <p><b>hh.mm.ss</b><br/>Specifies the time of day in the same way as for TIME data.</p> <p><b>nnnnnnnnnnnn</b><br/>Specifies a 0-digit to 12-digit number that represents the number of fractional seconds.</p> <p>For example, 2010-09-30-13.08.36.123456654321 is 1:08 P.M. and 36.123456654321 seconds on September 30, 2010, in the notation commonly used in the United States.</p>                                   |

Table 60. Formatting DATE, TIME, and TIMESTAMP data (continued)

| Data type                     | Form of the string                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TIMESTAMP WITH TIME ZONE data | <p><i>yyyy-mm-dd-hh-mm-ss-nnnnnnnnnnnzth:tm</i></p> <p><b>yyyy-mm-dd</b><br/>Specifies the date in the same way as for DATE data.</p> <p><b>hh.mm.ss</b><br/>Specifies the time of day in the same way as for TIME data.</p> <p><b>nnnnnnnnnnnn</b><br/>Specifies a 0-digit to 12-digit number that represents the number of fractional seconds.</p> <p><b>z</b><br/>A plus (+) or minus (-) sign that indicates the time zone offset relative to Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time (GMT).</p> <p><b>th</b><br/>A two-digit value representing the time zone hours.</p> <p><b>tm</b><br/>A two-digit value representing the time zone minutes.</p> <p>The valid range for the time zone portion of the format is from -24:00 to +24:00. To specify UTC, you can either specify a time zone of -0:00 or +0:00 or replace the time zone offset and its sign with an uppercase Z.</p> <p>For example, 2010-09-30-13.08.36.123456654321-08:00 indicates a time of 1:08 P.M. and 36.123456654321 seconds on September 30, 2010, in San Jose, California, in the United States. The timestamp 2010-09-30-13.08.36.123456654321Z indicates a time of 1:08 P.M. and 36.123456654321 seconds wherever UTC is in effect.</p> |

## Db2 exits for date and time data in TSO

When users create reports in QMF for TSO, they can specify the local format for DATE or TIME data by specifying the appropriate QMF edit code: TDL for dates; TTL for times. QMF does the formatting by calling the appropriate exit that is supplied with Db2: DSNXVDTX formats dates; DSNXVTMX formats times.

As shipped by Db2, these exits are stubs. These stubs are meant to be used when no local formats are defined; they do no formatting at all. You must replace them with your local copies of the exits for them to work properly.

Make your local copies of the exits available to QMF by placing their load libraries in the STEPLIB concatenation of your users' JCL. Ensure that this library is searched before the Db2 program library. If the program library is searched first, QMF loads and uses the two stubs that are supplied with Db2 instead. In the following STEPLIB statement example, the formatting routines are in the library XYZ.FORMAT, and the Db2 program library is DSN1110.SDSNLOAD.

```
//STEPLIB DD DSN=ISP.SISPLOAD,DISP=SHR
//          DD DSN=QMF1210.SDSQLOAD,DISP=SHR
//          DD DSN=XYZ.FORMAT,DISP=SHR          (local formatting library)
//          DD DSN=DSN1110.DSNLOAD,DISP=SHR    (DB2 program library)
//          DD DSN=GDDM.OSPID.SADMMOD,DISP=SHR
```

Figure 59. Making the edit routine available

If you choose to write an edit exit routine to carry out functions that are handled by the TTL and TDL edit codes, you cannot use TTL and TDL as the edit codes for those functions. Instead, use Uxxxx or Vxxxx edit codes to identify your local date and time exit routines.

## Edit routines for programming languages

You can write your edit exit routine to format the data described by your edit code in one of several languages. QMF provides both a standard interface to your edit exit routine and a sample edit exit program that you can use as a starting point for writing your own.

Different versions of a sample edit exit routine are placed in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The name of the sample routine varies by programming language and environment, as shown here:

| Language  | TSO, ISPF, and native z/OS batch | CICS     |
|-----------|----------------------------------|----------|
| COBOL     | DSQUXDTC                         | DSQUXCTC |
| PL/I      | DSQUXDTP                         | DSQUXCTP |
| Assembler | DSQUXDTA                         | DSQUXCTA |

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Writing an edit routine in High-Level Assembler

You can write an edit routine in Assembler for TSO, ISPF, native z/OS, and CICS.

### Writing an Assembler edit routine for TSO, ISPF, and native z/OS

To write an edit routine in Assembler for TSO, ISPF, and native z/OS you must understand the details that are specific to the language.

The QMF edit exit interface for Assembler consists of these parts:

- Interface control block

A sample Assembler DSECT for the interface control structure, DXEECS, is shipped with QMF as DXEECSA, located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language). The interface control structure defines the input fields for the edit interface, DSQUEDIT.

If you are using the example program (DSQUXDTA), a COPY statement that includes this DSECT is already included in the sample. If you are writing your own routine instead of starting with the example program, ensure that your program includes this COPY statement.

The interface control block contains the user's edit code, identifies the source data and the target location for the edited result, and provides a scratchpad area for the user edit routine's use. The control block is persistent between calls to the user edit routine. The scratchpad area is not modified by QMF after the initial invocation of the exit routine.

- Control program, which is shipped with QMF as DSQUXIA
- Your edit exit program, which is named DSQUXDT

The sample edit program for Assembler, DSQUXDTA, is located in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The sample program is commented so that you can modify it to suit your needs. If you plan to start with this example program, copy it to your private program source library and change its name to DSQUXDT.

The following figure shows the program structure of an Assembler edit exit routine for TSO, ISPF, or native z/OS.

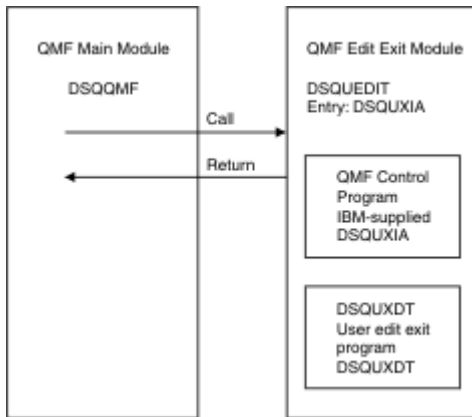


Figure 60. Program structure of an Assembler edit exit routine for TSO, ISPF, or native z/OS

### Statements for calling your program and returning control to QMF

The user edit program is called as a subroutine in TSO, ISPF, and native z/OS using a standard Assembler CALL statement. Linkage obeys the standard IBM calling conventions. On entry to your edit exit program, the following conditions exist:

- Register 1 contains the address of a standard parameter list.
- Register 13 contains the address of a standard SAVE area.
- Register 14 contains the caller's (QMF) return address.

The following figure shows an example of these conditions.

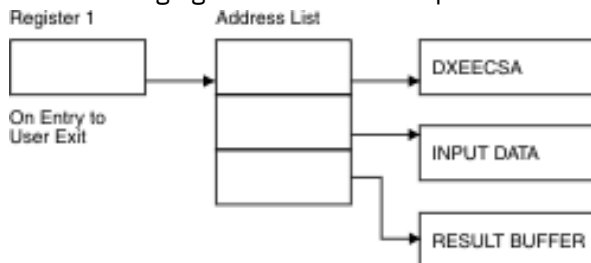


Figure 61. Conditions that exist on entry to the edit exit program in Assembler

Return control to QMF in the standard way by restoring registers to their values at the time of the call and then returning to the address in register 14.

In the example program, the addresses are placed in registers 10, 9, and 8 through the following statements:

```

ECSPTR   EQU R10
         L     ECSPTR,0(R1)
         USING DXEECS,ECSPTR
ECSINPTP EQU R9
         L     ECSINPTP,4(R1)
         USING ECSINPT,ECSINPTP
ECSRSLTP EQU R8
         L     ECSRSLTP,8(R1)
         USING ECSRSLT,ECSRSLTP
    
```

The USING statements refer to the DSECTs defined in DXEECSA. These define the three parameters and their input-field components.

Registers 10, 9, and 8 point, respectively, at the control block, the value to be formatted, and the storage reserved for the formatted results.

## Assembling and link-editing your program

When you assemble and link-edit your program, the QMF edit exit interface control block for Assembler, DXEECSA, must be available in a macro library. DXEECSA is located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language).

Create a new QMF edit exit module, DSQUEDIT, by including your edit program (DSQUXDT) with the control module, DSQUXIA, which is located in the QMF module library QMF1210.SDSQLOAD. DSQUXIA must be specified as the entry point.

Module DSQUEDIT can be executed in either 24-bit or 31-bit addressing mode. QMF runs in 31-bit addressing mode and automatically switches to 24-bit addressing mode if edit exit module DSQUEDIT has a 24-bit addressing mode. Thirty-one bit addressing mode is recommended.

These statements show examples for assembling and link-editing your job for TSO, ISPF, and native z/OS. Before running this job, create the load library referenced in the SYSLMOD statement; the user edit exit module will reside in this library.

```
//sampasm JOB
//STEP1 EXEC PROC=ASMACL
//* Provide access to QMF edit macro DXEECSA
//C.SYSLIB DD DSN=QMF1210.SDSQUSRE,DISP=SHR
//C.SYSIN DD *
.
.
.
Your program or copy of QMF sample DSQUXDTA
.
.
.
/*
//* Provide access to QMF interface module
//L.QMFLOAD DD DSN=QMF1210.SDSQLOAD,DISP=SHR
//L.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//L.SYSIN DD *
INCLUDE QMFLOAD(DSQUXIA)
ENTRY DSQUXIA
MODE AMODE(31) RMODE(31)
NAME DSQUEDIT(R)
/*
```

Figure 62. Example statements for assembling and link-editing an Assembler edit exit routine for TSO, ISPF, or native z/OS

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Related reference

### Fields of the interface control block

Use the fields of the interface control block to pass information to and from your exit routine.

## Writing an Assembler edit routine for CICS

To write an edit routine in Assembler for CICS you must understand the details that are specific to the language.

The QMF edit exit interface for Assembler in CICS consists of these parts:

- Interface control block between QMF and the user edit interface

An Assembler DSECT for the interface control block, DXEECS, is shipped with QMF as DXEECSA and is located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language).

If you are using the supplied sample program for CICS (DSQUXCTA), a COPY statement that includes this DSECT is already included in the sample. If you are writing your own program instead of starting with the supplied sample edit program, be sure to include the COPY statement in your program.

The interface control block contains the user's edit code, identifies the source data and the target location for the edited result, and provides a scratchpad area for the user edit routine's use. The control block is persistent between calls to the user edit routine. The scratchpad area is not modified by QMF after the initial invocation of the exit routine.

- CICS prolog and epilog macros, which are shipped with CICS as DFHEIENT and DFHEIRET
- CICS command interface modules, which are shipped with CICS as DFHEAI and DFHEAI0
- Your edit exit program, which is named DSQUEECIC

The supplied example edit program in Assembler, named DSQUXCTA, is located in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The example program is liberally commented; you can print it, browse it online, or modify it to meet your needs. If you plan to use this program, copy it to your private program source library and change its name to DSQUEECIC.

The program structure of an Assembler edit exit routine for CICS is illustrated here:

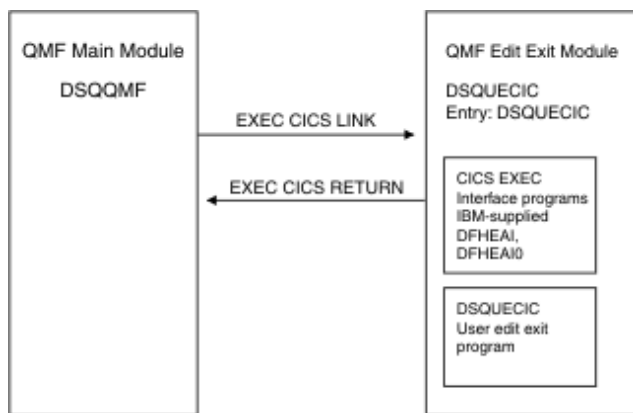


Figure 63. Program structure of an Assembler edit exit routine for CICS

### Statements for calling your program and returning control to QMF

The user edit program is called by using the standard CICS LINK command interface. Your program is executing on a different program level than the main QMF program. On entry to your edit exit program, the following conditions exist:

- Register 1 contains the address of a standard CICS parameter list suitable for processing by macros DFHEIENT and DFHEIRET that are supplied with CICS.

Here is the program flow:



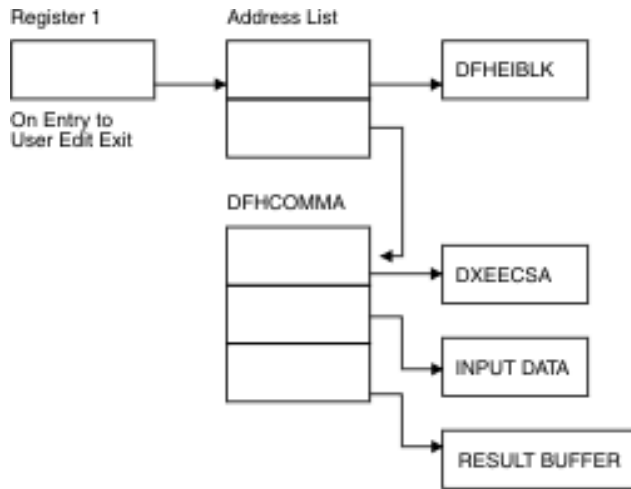


Figure 64. How a CICS Assembler edit routine interacts with QMF

- Register 13 contains the address of a standard CICS working storage area as described by macro DFHEISTG that is supplied with CICS.

Return control to QMF by using the standard CICS RETURN command (for example, EXEC CICS RETURN).

### Translating, assembling, and link-editing your program

You must translate your program using the CICS translator for Assembler. When you translate your program, CICS normally supplies the standard CICS prologue (DFHEIENT), which sets up addressability, saves registers in the standard CICS working storage area, and provides a standard CICS epilogue (DFHEIRET).

During assembly, QMF edit exit interface control block DXEECSA, located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language), and the CICS macro library must be available.

Create a new QMF edit exit module, DSQUEECIC, by including your edit program (DSQUEECIC) with EXEC CICS interface control modules DFHEAI and DFHEAIO, which are both located in the CICS module library as distributed by the CICS product. The EXEC CICS module DFHEAI must be the first module in the edit exit module and the entry point must be DSQUEECIC.

The module DSQUEECIC must be executable in 31-bit addressing mode.

The following statements are examples for translating, assembling, and link-editing your job for CICS. Before running this job, create the load library referenced in the SYSLMOD statement; the user edit exit module will reside in this library.

```

//SAMPASM JOB ...
//* Add a parameter PROGLIB to procedure DFHEITAL
//*      PROGLIB=&PROGLIB,
//TRNCOMLK EXEC PROC=DFHEITAL,PROGLIB=QMF1210.SDSQLOAD'
//TRN.SYSIN DD *
.
.
.
Your program or modified copy of QMF sample DSQUXCTA
.
.
.
/*
//* Provide access to QMF edit macro DXEECSA
//ASM.SYSLIB DD DSN=QMF1210.SDSQUSRE,DISP=SHR
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
INCLUDE SYSLIB(DFHEAI)
INCLUDE SYSLIB(DFHEAI0)
ORDER DFHEAI,DFHEAI0
ENTRY DSQUECIC
MODE AMODE(31) RMODE(31)
NAME DSQUECIC(R)
/*

```

Figure 65. Example statements for translating, assembling, and link-editing an Assembler edit exit routine for CICS

## Related tasks

### [Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Writing an edit routine in PL/I

You can write a QMF edit routine in PL/I.

### **The CEEPIPI Language Environment interface and PL/I edit routines for TSO, ISPF, or native z/OS**

Follow these instructions for using the CEEPIPI Language Environment<sup>®</sup> interface to write a QMF edit routine that runs in TSO, ISPF, or native z/OS.

This interface consists of the following parts:

- Interface control structure DXEECS

This control structure is shipped with QMF as DXEECS. The control structure is passed on all calls to the edit exit program. It contains status and communications information between QMF and the edit exit routine. It also contains information about the data to be formatted as well as a pointer to where the formatted result is stored.

For more information about the DXEECS control structure, see copy file DXEECS, which is in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language).

- A control program named DSQUXILE, which is shipped with QMF

DSQUXILE must be the main entry point to the QMF edit exit module, DSQUEDIT. DSQUXILE is in the QMF module library, QMF1210.SDSQLOAD. The DSQUXILE control program calls the CEEPIPI program to initialize the Language Environment and to terminate the Language Environment when the QMF session ends. The DSQUXILE control program calls the user exit program, DSQUXDT, by calling the CEEPIPI program and specifying execution of the DSQUXDT program.

- Your edit exit program

The example edit exit program for PL/I, named DSQUXDTP, is in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The example program is

heavily commented; it can be browsed online, printed, or modified to suit your needs. If you plan to use this program, copy it to your private program source library and change its name to DSQUXDT.

- Language Environment preinitialization service program, which is named CEEPIPI

The following figure shows the program structure of a PL/I edit exit routine that uses the CEEPIPI Language Environment interface.

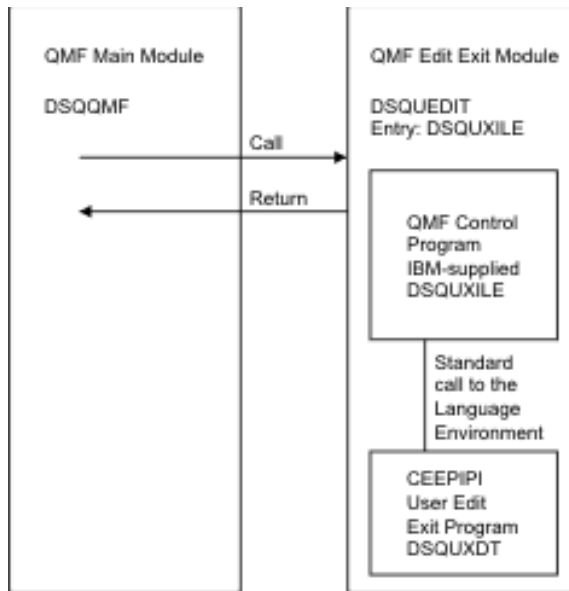


Figure 66. Program structure of a PL/I edit exit routine written with the CEEPIPI Language Environment interface

### Statements for calling your program and returning control to QMF

QMF calls your edit exit routine as a PL/I program by calling the CEEPIPI program and specifying execution of DSQUXDT. The following parameters are provided in the indicated order:

```
DSQUXDT:
  PROCEDURE (DXEECSF, ECSINPTF, ECSRSLTF) OPTIONS (REENTRANT);
```

Return control to QMF by using a standard RETURN statement.

### Compiling DSQUXDT

Compile your edit exit program by specifying REENTRANT as a procedure option. See [“Statements for calling your program and returning control to QMF” on page 269](#) for the proper syntax of the PROCEDURE statement.

During the compilation, QMF edit exit interface control block DXEECSF, in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language), must be available in the SYSLIB allocation.

### Link-editing your program

Create a new QMF edit exit module (DSQUEDIT) by including your edit program (DSQUXDT) with the control module (DSQUXILE), in QMF module library QMF1210.SDSQLOAD. DSQUXILE must be specified as the entry point.

Module DSQUEDIT can be executed in either 24-bit or 31-bit addressing mode. QMF runs in 31-bit addressing mode and automatically switches to 24-bit addressing mode if edit exit module DSQUEDIT has a 24-bit addressing mode.

Thirty-one bit addressing mode is recommended.

These statements show examples for assembling and link-editing your job for TSO, ISPF, or native z/OS. If your edit routine formats decimal floating-point data, you need to add the ARCH(7) and FLOAT(DFP) compiler options. Before you run this job, create the load library that is referenced in the SYSLMOD statement; the user edit exit module will reside in this library.

```
//samPLI JOB
//STEP1 EXEC PROC=IBMZCPL
//* Provide access to QMF edit macro DXEECS
//PLI.SYSLIB DD DSN=QMF1210.SDSQUSRE,DISP=SHR
//PLI.SYSIN DD *
.
.
.
Your PL/I edit exit program DSQUXDT
or copy of sample program DSQUXDTP
that has been renamed to DSQUXDT
.
.
.
If your edit routine formats decimal
floating-point data, add ARCH(7) and
FLOAT(DFP) compiler options here.
/*
/* Provide access to QMF interface module DSQUXILE
//LKED.QMFLOAD DD DSN=QMF1210.SDSQLOAD,DISP=SHR
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
INCLUDE QMFLOAD(DSQUXILE)
ENTRY DSQUXILE
MODE AMODE(31) RMODE(ANY)
NAME DSQUEDIT(R)
/*
```

Figure 67. Example statements for compiling and link-editing a PL/I routine that was written with the CEEPIPI Language Environment interface

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Related reference

### Fields of the interface control block

Use the fields of the interface control block to pass information to and from your exit routine.

## Writing a PL/I edit routine for CICS

To write an edit routine in PL/I for CICS you must understand the details that are specific to the language.

The QMF edit exit interface for PL/I in CICS consists of these parts:

- A PL/I data structure

This data structure is shipped with QMF as DXEECS and is provided in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language). Include this structure in your program.

The data structure is passed on all calls to the edit exit program. It contains status and communications information between QMF and the edit exit routine. It also contains information about the data to be formatted as well as a pointer to where the formatted result is stored.

- CICS command-level interface

QMF uses the CICS command-level interface to invoke the QMF edit exit module, DSQUEECIC. Your completed edit exit module must issue an EXEC CICS RETURN statement to pass control back to QMF.

- Your edit exit program, which is named DSQUEECIC

When QMF is installed, the QMF edit exit program is installed with a program language of Assembler; the supplied example edit program for PL/I in CICS is named DSQUXCTP and is located in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language).

The example program is liberally commented; you can print it, browse it online, or modify it to meet your needs. If you plan to use this program, copy it to your private program source library and change its name to DSQUEECIC.

When QMF is installed, the CICS program definition resource is configured to Assembler. To use the PL/I edit exit program, you must reconfigure the CICS program definition to PL/I.

Here is the program structure of a PL/I edit exit routine in CICS:

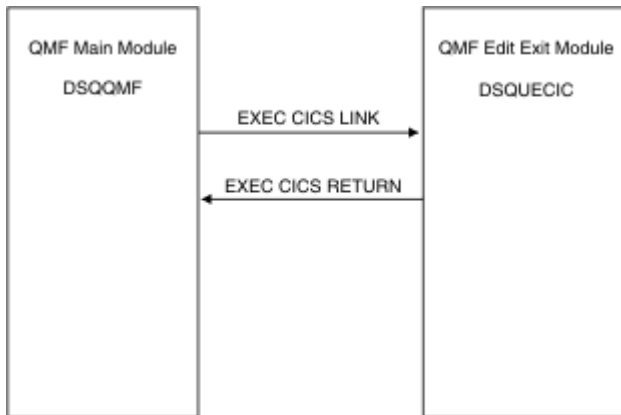


Figure 68. Program structure for PL/I edit exit routine in CICS

### Statements for calling your program and returning control to QMF

The user edit program is called by using the standard CICS LINK command interface. Your program is executing on a different program level than the main QMF program. The user edit program must be translated using the CICS translator for PL/I.

QMF provides addresses to the user edit routine control block (DXEECS), as well as input data and output data, in the CICS communications area, DFHCOMMAREA, as shown here.

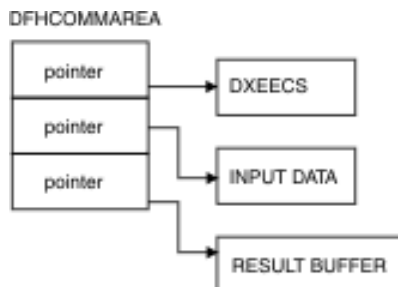


Figure 69. How DFHCOMMAREA, the CICS communications area, works

After translation, the CICS translator provides a procedure statement that describes the CICS environment block, DFHEIBLK. Provide a parameter that points to the CICS communications block, DFHCOMMAREA, as in the following example:

```

DSQUEECIC:
  PROCEDURE(DFHCOMMP) OPTIONS(REENTRANT,MAIN);
  
```

Also provide your own description of the DFHCOMMAREA in the PL/I program, as shown here:

```

/*****/
/* CICS DFHCOMMAREA DESCRIPTION OF EDIT EXIT PARAMETERS */
/*****/
DECLARE
  DFHCOMP PTR;
DECLARE
  1 DFHCOMM BASED(DFHCOMP),
    02 DFHCOMM_ECSPTR PTR,
    02 DFHCOMM_INPTR PTR,
    02 DFHCOMM_OUTPTR PTR;

```

Figure 70. Description of edit exit parameters in DFHCOMMAREA, the CICS communications area

To provide addressability to the user edit routine control block, DXEECS, as well as input data area ECSINPT and result data area ECSRSLT, set the addresses of these data areas to the values located in DFHCOMMAREA, as shown in here:

```

ECSPTR   = DFHCOMM_ECSPTR   /* ADDRESS OF DXEECS:
                             EDIT CODE SPECIFICATIONS      */
ECSINPTP = DFHCOMM_INPTR   /* ADDRESS OF INPUT DATA      */
ECSRSLTP = DFHCOMM_OUTPTR  /* ADDRESS OF RESULT AREA     */

```

Figure 71. Providing addressability to the DXEECS control block

Return control to QMF using a standard CICS RETURN command such as the following:

```
EXEC CICS RETURN;
```

### Translating, compiling, and link-editing your program

Translate your program using the CICS translator for PL/I. During translation, CICS supplies an input parameter and data-structure definition for the CICS environment control block (EIB).

The QMF edit exit interface control block, DXEECS, is located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language); it must be available in a macro library during the compilation.

Specify PL/I compiler options PP(CICS), SYSTEM(CICS), and RENT.

Create a new QMF edit exit module, DSQUEECIC, that has a run mode (RMODE) and address mode (AMODE) of 31-bit.

The following statements are examples for translating, compiling, and link-editing your job for CICS. If your edit routine formats decimal floating-point data, you need to add the ARCH(7) and FLOAT(DFP) compiler options. Before running this job, create the load library referenced in the SYSLMOD statement; the user edit exit module will reside in this library.

```

//SAMPLI    JOB    .....
//COMLK     EXEC  PROC=IBMZCPL
//PLI.SYSLIB DD   DSN=QMF1210.SDSQUSRE,DISP=SHR
//PLI.SYSIN  DD   *
.
.
Your program or modified copy of QMF sample DSQXCTP
that has been renamed to DSQUEECIC
.
.
Include compiler options: PP(CICS),SYSTEM(CICS) and RENT.
If your edit routine formats decimal
floating-point data, add ARCH(7) and
FLOAT(DFP) compiler options.
.
.
/*
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD   *
MODE AMODE(31),RMODE(31)
NAME DSQUEECIC(R)
/*

```

Figure 72. Example statements for translating, compiling, and link-editing your edit exit in PL/I for CICS

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Writing an edit routine in COBOL

You can write a QMF edit routine in COBOL.

### The IGZERRE interface and COBOL edit routines for TSO, ISPF, or native z/OS

Follow these instructions for using the IGZERRE interface to write a COBOL edit routine that runs in TSO, ISPF, or native z/OS.

This interface consists of the following parts:

- Interface control structure DXEECS

This control structure is shipped with QMF. Include it in your program.

The control structure is passed on all calls to the edit exit program. It contains status and communications information between QMF and the edit exit routine. It also contains information about the data to be formatted as well as a pointer to where the formatted result is stored. For more information about the DXEECS control structure, see copy book DXEECS, which is provided in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language).

- A control program named DSQUXIC

This control program is shipped with QMF. It must be the main entry point to the QMF edit exit module, DSQUEDIT. DSQUXIC is in the QMF module library, QMF1210.SDSQLOAD. The DSQUXIC control program calls the IGZERRE program to initialize the COBOL environment and to terminate the COBOL environment when the QMF session ends. The DSQUXIC control program calls user exit program DSQUXDT as a COBOL program.

- Your edit exit program, which is named DSQUXDT

The example edit exit program in COBOL, named DSQUXDTC, is in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The example program is

liberally commented; it can be browsed online, printed, or modified to suit your needs. If you plan to use this program, copy it to your private program source library and change its name to DSQUXDT.

The following figure shows the program structure of a COBOL edit exit routine

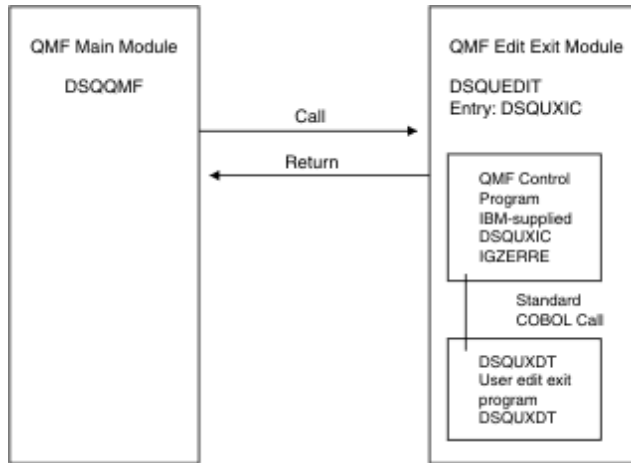


Figure 73. Program structure of a COBOL edit exit routine

### Statements for calling your program and returning control to QMF

QMF calls your edit exit routine as a COBOL program that uses a standard COBOL CALL statement. The following parameters are provided in the indicated order:

```

PROCEDURE DIVISION
  USING DXEECS, ECSINPT, ECSRSLT.
  
```

Return control to QMF by using a standard GOBACK statement.

### Compiling and link-editing your program

Compile the program with the LIB, RENT, QUOTE, and NODYNAM options. The user edit routine control block, DXEECS, uses quotes as literal delimiters. You must use the QUOTE compiler option if you use the DXEECS control block as distributed by IBM.

During the compilation, QMF edit exit interface control block DXEECS, in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language), must be available in the SYSLIB allocation.

After you compile DSQUXDT, place the resulting load module in the QMF1210.SDSQLOAD library.

You create a new QMF edit exit module (DSQUEDIT) by including your edit exit program (DSQUXDT) with the control module (DSQUXIC). DSQUXIC is in the QMF module library, QMF1210.SDSQLOAD. DSQUXIC must be specified as the entry point. Module DSQUEDIT can be executed in either 24-bit or 31-bit addressing mode. QMF runs in 31-bit addressing mode and automatically switches to 24-bit addressing mode if edit exit module DSQUEDIT has a 24-bit addressing mode. 31-bit addressing mode is recommended.

These statements show examples for compiling and link-editing your applications that run in TSO, ISPF, or native z/OS. Before you run this job, create the load library that is referenced in the SYSLMOD statement; the user edit exit module will reside in this library.



```

//samCOBOL JOB
//STEP1 EXEC PROC=IGYWCL,PARM='LIB,RENT,NODYNAM,QUOTE'
//* Provide access to QMF edit macro DXEECS
//COBOL.SYSLIB DD DSN=QMF1210.SDSQUSRE,DISP=SHR
//COBOL.SYSIN DD *
.
.
Your COBOL edit exit program DSQUXDT or a copy of
sample program DSQUXDTC that has been renamed to
DSQUXDT
.
.
/*
/* Provide access to QMF interface module DSQUXIC
//LKED.QMFLOAD DD DSN=QMF1210.SDSQLOAD,DISP=SHR
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
INCLUDE QMFLOAD(DSQUXIC)
INCLUDE SYSLIB(IGZERRE)
ENTRY DSQUXIC
MODE AMODE(31) RMODE(ANY)
NAME DSQUEDIT(R)
/*

```

Figure 74. Example statements for compiling and link-editing an edit exit that was written in COBOL and that uses the IGZERRE interface

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### The CEEPIPI Language Environment interface and COBOL edit routines for TSO, ISPF, or native z/OS

Follow these instructions for using the CEEPIPI Language Environment interface to write a QMF edit routine in COBOL that runs in TSO, ISPF, or native z/OS.

This interface cannot be used for edit exit programs that run in QMF for CICS.

The QMF edit exit interface in COBOL for TSO, ISPF, or native z/OS consists of these parts:

- Interface control structure DXEECS

This control structure is shipped with QMF as DXEECS. Include this structure in your program. The information in this control structure is passed on all calls to the edit exit program. It contains status and communications information between QMF and the edit exit routine. It also contains information about the data to be formatted as well as a pointer to where the formatted result is stored.

For details about the DXEECS interface control structure, see the copy book DXEECS, which is in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language).

- Control program, which is provided by IBM and shipped with QMF as DSQUXILE

The QMF control program, DSQUXILE, is provided by IBM and must be the main entry point to the QMF edit exit module, DSQUEDIT. DSQUXILE is in the QMF module library QMF1210.SDSQLOAD. The DSQUXILE control program calls the CEEPIPI program to initialize the Language Environment and to terminate the Language Environment when the QMF session ends. The DSQUXILE control program calls the user exit program, DSQUXDT, by calling the CEEPIPI program and specifying execution of the DSQUXDT program.

- Your edit exit program, which is named DSQUXDT

The example edit exit program in COBOL, named DSQUXDTC, is in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The example program is

heavily commented; it can be browsed online, printed, or modified to suit your needs. If you plan to use this program, copy it to your private program source library and change its name to DSQUXDT.

- Language Environment preinitialization service program, which is named CEEPIPI

The following figure shows the program structure of a COBOL edit exit routine that uses the CEEPIPI interface:

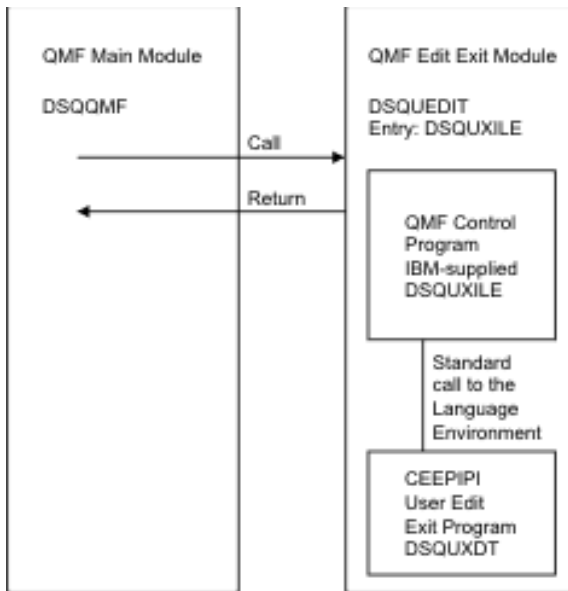


Figure 75. Program structure of a COBOL edit exit routine that uses the CEEPIPI interface

The edit control block (DXEECS) and the sample COBOL program (DSQUXCTC), as shipped with QMF, use quotes (") to delimit literals. If your site or program uses apostrophes (') instead, you have to change DXEECS or copy the structure to your program, changing quotes to apostrophes.

### Statements for calling your program and returning control to QMF

QMF calls your edit exit routine as a COBOL program by calling the CEEPIPI program and specifying execution of DSQUXDT. The following parameters are provided in the indicated order:

```
PROCEDURE DIVISION
    USING DXEECS, ECSINPT, ECSRSLT.
```

Use a standard GOBACK statement to return control from your COBOL edit exit program, DSQUXDT, to QMF.

### Compiling DSQUXDT

During the compilation, QMF edit exit interface control block DXEECS, in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language), must be available in the SYSLIB allocation.

Compile the program with the LIB, RENT, QUOTE, and NODYNAM options. DXEECS uses quotes as literal delimiters. You must use the QUOTE compiler option if you use the DXEECS control block as distributed by IBM.

### Link-editing your program

You create a new QMF edit exit module (DSQUEDIT) by including your edit exit program (DSQUXDT) with the control module DSQUXILE. DSQUXILE is in the QMF module library (QMF1210.SDSQLOAD). DSQUXILE must be specified as the entry point.

Module DSQUEDIT can be executed in either 24-bit or 31-bit addressing mode. QMF runs in 31-bit addressing mode and automatically switches to 24-bit addressing mode if the edit exit module DSQUEDIT has a 24-bit addressing mode; 31-bit addressing mode is recommended.

These statements show examples for compiling and link-editing your job for TSO, ISPF, or native z/OS. Before you run this job, create the load library that is referenced in the SYSLMOD statement; the user edit exit module will reside in this library.

```
//samCOBOL JOB
//STEP1 EXEC PROC=IGYWCL,PARM='LIB,RENT,NODYNAM,QUOTE'
//* Provide access to QMF edit macro DXEECS
//COBOL.SYSLIB DD DSN=QMF1210.SDSQUSRE,DISP=SHR
//COBOL.SYSIN DD *
.
.
.
Your COBOL edit exit program DSQUXDT
.
.
.
/*
/* Provide access to QMF interface module DSQUXILE
//LKED.QMFLOAD DD DSN=QMF1210.SDSQLOAD,DISP=SHR
/* Provide access to the LE program library
//LKED.SYSLIB DD ...
// DD DSN=SYS1.SCEELKED,DISP=SHR
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
INCLUDE QMFLOAD(DSQUXILE)
ENTRY DSQUXILE
MODE AMODE(31) RMODE(ANY)
NAME DSQUEDIT(R)
/*
```

Figure 76. Example statements for compiling and link-editing an edit exit routine that was written in COBOL and that uses the CEEPIPI interface

## Related tasks

### [Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Related reference

### [Fields of the interface control block](#)

Use the fields of the interface control block to pass information to and from your exit routine.

## Writing a COBOL edit routine for CICS

To write an edit routine in COBOL for CICS you must understand the details that are specific to the language.

The edit exit interface for COBOL in CICS consists of these parts:

- Interface control block DXEECS

For details on this control block, see the copy book DXEECS, which is shipped with QMF and is located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language). Include this copy book in your program.

- CICS command-level interface

QMF uses the CICS command-level interface to invoke the QMF edit exit module, DSQUEECIC. Your completed edit exit module must issue an EXEC CICS RETURN statement to pass control back to QMF.

- Your edit exit program, which must be named DSQUEECIC

When QMF is installed, the QMF edit exit program is installed with a program language of Assembler. To use the COBOL edit exit program, you must change the program language of module DSQUEECIC to COBOL in the CICS program resource control table.

The IBM-supplied example edit program in COBOL, named DSQUXCTC, is located in the QMF1210.SDSQSAP $n$  library (where  $n$  is a 1-character identifier that represents your national language). The example program is liberally commented; it can be browsed online, printed, or modified to suit your needs. To use the example program, copy it to your private program source library and change its name to DSQUXCT.

The following figure shows the structure of a COBOL edit exit routine in CICS.

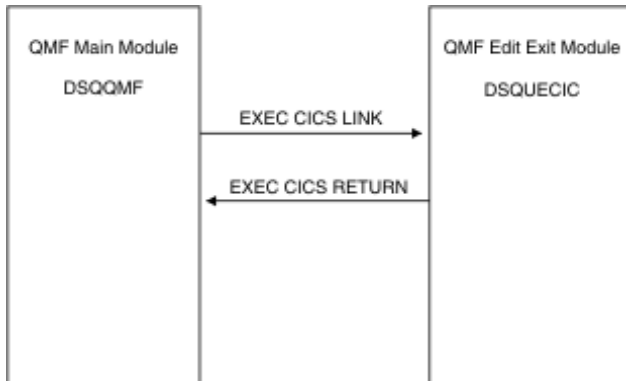


Figure 77. Program structure for a COBOL edit exit routine in CICS

### Statements for calling your program and returning control to QMF

The user edit program is called by using the standard CICS LINK command interface. Your program is executing on a different program level than the main QMF program. The interface control block between QMF and the user edit interface (DSQUEDIT) is DXE ECS. It contains the user's edit code and provides a scratchpad area for the user edit routine's use. The control block is persistent between calls to the user edit routine. The scratchpad area is not modified by QMF after the initial invocation of the exit routine.

The user edit program must be translated using the CICS translator for COBOL. The CICS communications area, DFHCOMMAREA, is used to provide addresses to the user edit routine program parameters (DXE ECS, input data, and output data) as shown here.

DFHCOMMAREA

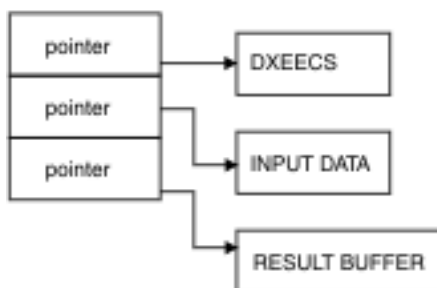


Figure 78. Providing addressability for your edit exit routine in CICS using the DFHCOMMAREA

After translation, the CICS translator provides a procedure statement that describes the CICS environment block, DFHEIBLK, and the CICS communications block, DFHCOMMAREA, as in the following example:

```
PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA.
```

As shown in the previous figure, QMF provides addresses to the user edit routine control block DXEECS, input data, and output data in the CICS communications area, DFHCOMMAREA. Provide your own description of the DFHCOMMAREA in the COBOL program linkage section as follows:

```
LINKAGE SECTION.  
  01 DFHCOMMAREA.  
    02 ECSADR  POINTER.  
    02 ECSINADR POINTER.  
    02 ECSRLADR POINTER.
```

To provide addressability to the user edit routine control block, DXEECS, input data area ECSINPT, and result data area ECSRSLT, set the addresses of these data areas to the values located in the DFHCOMMAREA as in the following example:

```
SETUP SECTION.  
  
  SET ADDRESS OF DXEECS  TO ECSADR.  
  SET ADDRESS OF ECSINPT TO ECSINADR.  
  SET ADDRESS OF ECSRSLT TO ECSRLADR.
```

Return control to QMF by using a standard CICS RETURN command such as the following:

```
EXEC CICS  
      RETURN  
END-EXEC.
```

### Translating, compiling, and link-editing your COBOL program

Translate your program using the CICS translator for COBOL. When you translate your program, CICS normally supplies the standard procedure and linkage sections. Replace the standard CICS communications area, DFHCOMMAREA, by providing a structure as specified in the linkage section example in [“Statements for calling your program and returning control to QMF”](#) on page 278.

QMF edit exit interface control block DXEECS, located in the QMF1210.SDSQUSR $n$  library (where  $n$  is a 1-character identifier that represents your national language), must be available in a macro library during the compile.

Specify COBOL compiler options RENT, RES, and NODYNAM, and runtime options NOSTAE and NORTEREUS.

DXEECS uses quotes as literal delimiters. You must use the QUOTE compiler option if you use the DXEECS control block as distributed by IBM.

You create a new QMF edit exit module, DSQUECIC, by including your edit exit program (DSQUXCTC) with the EXEC CICS interface control module supplied by CICS. Module DSQUECIC must be executable in 31-bit addressing mode.

These statements show examples for translating, compiling, and link-editing your job for CICS. Before running this job, create the load library referenced in the SYSLMOD statement; the user edit exit module will reside in this library.

---

```

//SAMCOBOL JOB ...
//* Add a parameter PROGLIB to procedure DFHYITVL
//*   PROGLIB=&PROGLIB,
//TRNCOMLK EXEC PROC=DFHYITVL,PROGLIB=QMF1210.SDSQLOAD',
//   PARM.TRN='QUOTE',
//   PARM.COB='RENT,NODYNAM,OBJECT,LIB,LIST,MAP,QUOTE'
//TRN.SYSIN DD *
.
.
Your program or modified copy of QMF sample DSQUXCTC
.
.
/*
/* Provide access to QMF edit macro DXEECS
//COB.SYSLIB DD DSN=QMF1210.SDSQUSRE,DISP=SHR
//COB.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
ENTRY DSQUEECIC
MODE AMODE(31) RMODE(31)
NAME DSQUEECIC(R) /*

```

*Figure 79. Example statements for translating, compiling, and link-editing a COBOL edit exit program for CICS*

---

## **Related tasks**

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

---

## Chapter 17. Controlling QMF resources

A governor exit routine helps you limit end-user activity and control use of system resources at your site.

**Note:** This topic contains General Use Programming Interface and Associated Guidance Information.

Several governing tools are available for use with QMF:

- You can use governing functions provided with QMF for TSO and CICS.
- You can use the QMF High Performance Option/Manager (HPO/Manager) to manage and control QMF session activity. HPO/Manager provides a real-time user interface for QMF session activity and a query analyzer that estimates a query's resource use before it is run. The HPO/Manager replaces the QMF governor.
- You can use the Db2 resource limit facility to govern QMF operations.

---

### The default governor exit routine provided with QMF for TSO and CICS

This default governor routine provides default resource limits. For example, you can limit the number of rows a user can retrieve from the database or the amount of time spent running QMF commands.

The governor exit routine supplied for use with TSO, ISPF, and native z/OS (DSQUEGV1) provides options for controlling how many rows a user can retrieve from the database as well as the processor time that is spent running a QMF command. The governor exit routine supplied for CICS (DSQUEGV3) controls how many rows a user can retrieve from the database.

#### How a governor exit routine controls resources

The governor uses two types of information to control resources.

These types of information are used by the governor to control resources:

- Information about the resource limits you set for a user, defined in a table called Q.RESOURCE\_TABLE.
- Information about the state of the user's session, which tells the governor how close the user's activity is coming to the resource limits defined for the resource group the user is in.

This information is passed to the governor exit routine in the control blocks DXEGOVA and DXEXCBA. These control blocks cannot be updated if you are using the default governor exit routine; for information on how to modify the default governor or write your own routine, see [“Modifying the default governor exit routine or writing your own routine”](#) on page 298.

#### What happens when you reach a resource limit

When the resource control information QMF passes to the governor exit routine indicates that a resource limit has been reached, the default governor exit routine calls the QMF cancellation service to cancel the QMF activity the user tried to perform.

If you use the default limits for number of rows, the governor exit routine also displays a warning before canceling the activity, as shown in [Figure 81 on page 285](#). You can also activate this warning if you are not using the default values for the number of rows retrieved.

The default governor exit routine resets its count of the number of rows upon returning control to QMF, so that the number of rows is not cumulative across calls to the governor.

#### Related tasks

[Defining your own resource limits](#)

To set your own resource limits for the number of rows that are retrieved from the database, you can add a resource group to the resource control table.

### How the governor knows what the resource limits are

Each row of the Q.RESOURCE\_TABLE supplied by IBM contains resource control information.

Each row of the Q.RESOURCE\_TABLE contains these values:

- The name of a resource group (RESOURCE\_GROUP), which characterizes one or more users whose activities you want to govern in the same manner.
- The name of the resource (RESOURCE\_OPTION) that you want to limit for the group of users named in RESOURCE\_GROUP.
- Values (INTVAL, FLOATVAL, or CHARVAL) that define the limits for the resource option. Resource options can have integer values, floating-point values, or character values.

The following table describes each of the columns in Q.RESOURCE\_TABLE. The table has the index Q.RESOURCE\_INDEX. Keyed columns are RESOURCE\_GROUP and RESOURCE\_OPTION.

*Table 62. Structure of the Q.RESOURCE\_TABLE table*

| Column name     | Data type and length | Nulls allowed? | Function/values                                                                                                                                                                                                                                                                            |
|-----------------|----------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RESOURCE_GROUP  | CHAR(16)             | No             | Contains the name of the resource group. Update the RESOURCE_GROUP field of the user's row in Q.PROFILES to activate governing for that user.<br><br>The resource group in use for the user's QMF session is recorded in the DSQAP_RESOURCE_GRP global variable.                           |
| RESOURCE_OPTION | CHAR(16)             | No             | Your own name for a resource that you want to monitor.                                                                                                                                                                                                                                     |
| INTVAL          | INTEGER              | Yes            | Reflects the resource limit for resource options that have integer values. For example, the number of rows retrieved from the database is a resource that has an integer value.                                                                                                            |
| FLOATVAL        | FLOAT                | Yes            | Reflects the resource limit for resource options that have floating-point values. FLOATVAL is null for the default governor.                                                                                                                                                               |
| CHARVAL         | VARCHAR(80)          | Yes            | Reflects the resource limit for resource options that have character values. For example, you might establish a DAY_OF_WEEK resource option and assign MONDAY to CHARVAL so that QMF users can log on to QMF only on Mondays. CHARVAL is used as a comment column in the default governor. |

The following two tables show the default rows of the Q.RESOURCE\_TABLE as it is shipped by IBM. The default table includes a predefined resource group named SYSTEM.

This next table shows rows that are common to all environments.

*Table 63. Default resource group and options for the default governor exit. These options are common to all environments.*

| RESOURCE_GROUP | RESOURCE_OPTION | INTVAL | FLOATVAL | CHARVAL                             |
|----------------|-----------------|--------|----------|-------------------------------------|
| SYSTEM         | SCOPE           | -      | -        | Indicate whether governor is active |



Table 63. Default resource group and options for the default governor exit. These options are common to all environments. (continued)

| RESOURCE_GROUP | RESOURCE_OPTION | INTVAL | FLOATVAL | CHARVAL                                     |
|----------------|-----------------|--------|----------|---------------------------------------------|
| SYSTEM         | ROWPROMPT       | 25000  | -        | Prompt user after fetching 25,000 rows      |
| SYSTEM         | ROWLIMIT        | 100000 | -        | Cancel activity after fetching 100,000 rows |

And this table shows the rows of Q.RESOURCE\_TABLE that apply specifically to TSO, ISPF, and native z/OS batch environments.

Table 64. Additional default resource options that are only available in TSO, ISPF, or native z/OS batch environments

| RESOURCE_GROUP | RESOURCE_OPTION | INTVAL | FLOATVAL | CHARVAL                                 |
|----------------|-----------------|--------|----------|-----------------------------------------|
| SYSTEM         | TIMECHECK       | 900    | -        | 15 minute interval between time check   |
| SYSTEM         | TIMEPROMPT      | 360    | -        | Prompt user after 6 minutes of CPU time |
| SYSTEM         | TIMELIMIT       | 1440   | -        | Cancel after 24 minutes of CPU time     |

The resource options shown in the preceding two tables have the following meanings:

#### SCOPE

Used to activate governing:

- A value of zero in the INTVAL column of Q.RESOURCE\_TABLE activates governing for a particular resource group.
- Any non-zero value for SCOPE, including a null (the default), deactivates governing for the resource group.

#### ROWPROMPT = 25000

Warns the user when 25,000 database rows have been retrieved.

#### ROWLIMIT = 100000

If the user decides to continue when warned, the governor exit routine cancels data retrieval activities after 100,000 rows are retrieved. (Retrieval is for FETCH only.)

ROWLIMIT is dependent on the buffer size; therefore, more than 100,000 rows can be retrieved if the buffer holds a number of rows not divisible by 100,000.

The three additional options provided for TSO, ISPF, and native z/OS batch are:

#### TIMECHECK = 900

TIMECHECK is specified in seconds of real time. Thus, the value 900 specifies 15 minutes of real time between time checks or prompting or canceling.

#### TIMEPROMPT = 360

TIMEPROMPT is specified in seconds of processor time. Processor time refers to the jobstep time plus the SBR (Service Request Block) time. Thus, the value 360 warns the user when 6 minutes of processor time have elapsed. Evaluated after a TIMECHECK interval is processed.

#### TIMELIMIT = 1440

TIMELIMIT is specified in seconds of processor time. Thus, if the user decides to continue when warned, the governor exit routine cancels the command after 1440 seconds, or 24 minutes, of processor time have elapsed. TIMELIMIT is checked at TIMECHECK intervals; therefore, more than 24 minutes of processor time can elapse if the TIMECHECK interval is set at an interval not divisible by 24. TIMELIMIT is evaluated after a TIMECHECK interval is processed.

## Related concepts

How the governor knows when you reach a resource limit

The governor exit routine compares information from a user's row in the Q.PROFILES table against a value in the Q.RESOURCE\_TABLE.

## How the governor knows when you reach a resource limit

The governor exit routine compares information from a user's row in the Q.PROFILES table against a value in the Q.RESOURCE\_TABLE.

IBM supplies a view of Q.RESOURCE\_TABLE called Q.RESOURCE\_VIEW. The view includes all five columns of Q.RESOURCE\_TABLE. Each time QMF calls the governor exit routine, QMF passes to the routine the resource control information stored in Q.RESOURCE\_VIEW according to the resource group to which the user belongs. To determine the resource group, QMF checks the value of the RESOURCE\_GROUP column of the user's row in the Q.PROFILES table and checks Q.RESOURCE\_VIEW for a matching value. The governor exit routine uses this resource information to help determine when the user reaches a resource limit.

QMF uses two control blocks, DXEGOVA and DXEXCBA, to pass information to the governor exit routine. The DXEGOVA control block contains information from Q.RESOURCE\_VIEW about the limits you set for each user. The DXEXCBA control block contains information about the activities the user is performing in the current QMF session, which tells the governor how close the user is coming to the resource limits. These control blocks cannot be modified if you are using the default governor exit routine.

QMF makes function calls to the governor exit routine at a number of different points within the QMF session.

## Related concepts

How the governor knows what the resource limits are

Each row of the Q.RESOURCE\_TABLE supplied by IBM contains resource control information.

Points at which QMF calls the governor

Function calls to the governor exit routine either precede or follow a specific type of QMF activity.

## Related tasks

Modifying the default governor exit routine or writing your own routine

If you decide to govern resources other than the number of rows returned from the database or the processor time expired, you need to modify the default governor exit routine or write your own.

## Resource limits with the default governor exit

The governor exit routine supplied for use with TSO, ISPF, and native z/OS (DSQUEGV1) provides options for controlling how many rows a user can retrieve from the database as well as the processor time that is spent running a QMF command. The governor exit routine supplied for CICS (DSQUEGV3) controls how many rows a user can retrieve from the database.

### Default resource limits

The default governor monitors and controls the number of rows returned from the database. In addition, the governor running under TSO, ISPF, and native z/OS has values for the time spent running a QMF command.

The default governor exit routine is shipped with two predefined values for the number of rows:

- Number of rows at which the user is warned that a resource limit is approaching

A prompt panel warns users when the number of rows retrieved reaches 25,000, at which time the user sees the message in [Figure 80 on page 284](#).

```
DSQUn00 QMF governor prompt:  
Command has fetched 25,000 rows of data.
```

```
==> To continue QMF command press the "ENTER" key.  
==> To cancel QMF command type "CANCEL" then press the "ENTER" key  
==> To turn off prompting type "NOPROMPT" then press the "ENTER" key
```

*Figure 80. Message displayed when a resource limit is approaching*

**Important:** Database activity is not suspended when a cancellation prompt is displayed. Db2 continues to fetch rows and use processor time.

- Number of rows at which the QMF command is canceled

The default QMF governor cancels data retrieval when 100,000 rows have been retrieved (if the user presses the Enter key in response to the message in [Figure 80 on page 284](#)). When the governor cancels data retrieval, the user sees the message shown in [Figure 81 on page 285](#).

Row limit exceeded! Your command canceled by QMF governor.

*Figure 81. Message displayed when a resource limit is exceeded*

When running a procedure, you might get a message that your procedure was canceled, rather than the message in [Figure 81 on page 285](#). For example, if your procedure contains a command that requires that the report be completed (such as ERASE), you receive the message shown in [Figure 82 on page 285](#).

Procedure canceled.

*Figure 82. Message displayed when a procedure is canceled*

Users using the SYSTEM profile are already set up to use these default values of 25,000 and 100,000 rows.

TSO, ISPF, and native z/OS have two additional predefined values (a time limit and a time prompt value) for the time spent running a QMF command:

- A time prompt panel warns users when the processor time for the cycle reaches six minutes, at which time the user sees the message shown in [Figure 83 on page 285](#).

DSQUn00 QMF governor prompt:  
Command has executed for 6 minutes

==> To continue QMF command press the "ENTER" key.  
==> To cancel QMF command type "CANCEL" then press the "ENTER" key  
==> To turn off prompting type "NOPROMPT" then press the "ENTER" key

*Figure 83. Message displayed when a resource limit is approaching*

- A time limit value cancels the command when 24 minutes of processor time are used during the cycle.

### Activating the default limits

You can set up the governor exit routine to warn a user when the number of rows retrieved from the database reaches 25,000 and to cancel QMF activity when the number of rows retrieved reaches 100,000.

### About this task

If you want to define row limits other than the defaults of 25,000 and 100,000, read [“How a governor exit routine controls resources” on page 281](#). Then see the procedure in [“Defining your own resource limits” on page 286](#).

### Procedure

1. Enter the statement shown in [Figure 84 on page 285](#) on the **SQL Query** panel and issue the RUN QUERY command to update the Q.RESOURCE\_VIEW table.

```
UPDATE Q.RESOURCE_VIEW
SET INTVAL=0
WHERE RESOURCE_OPTION='SCOPE' AND
RESOURCE_GROUP='SYSTEM'
```

*Figure 84. Activating default values for the governor supplied by IBM*

2. Unless you have started QMF with a value of TSOID for the DSQSPRID parameter, set a value of SYSTEM for the RESOURCE\_GROUP field of the user's profile.

**Important:** Always specify a value for the TRANSLATION column, or you might change more rows in the Q.PROFILES table than you intend.

For example, the UPDATE statements in Table 65 on page 286 activate default values for user JONES (using English QMF) and user SCHMIDT (using German QMF).

If you have started QMF with a DSQSPRID value of TSOID, the resource group name is the user's TSO user ID.

The resource group in use for the user's QMF session is recorded in the DSQAP\_RESOURCE\_GRP global variable.

| <i>Table 65. Updating a user's resource group to use the default resource limits</i>                       |                                                                                                              |
|------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <b>Base QMF (English)</b>                                                                                  | <b>German NLF</b>                                                                                            |
| <pre>UPDATE Q.PROFILES SET RESOURCE_GROUP = 'SYSTEM' WHERE CREATOR='JONES' AND TRANSLATION='ENGLISH'</pre> | <pre>UPDATE Q.PROFILES SET RESOURCE_GROUP = 'SYSTEM' WHERE CREATOR='SCHMIDT' AND TRANSLATION='DEUTSCH'</pre> |

### What to do next

Instruct users to reconnect to the database to activate the new values. This can be done with a QMF CONNECT command, or they can end the current QMF session and begin another to activate the new resource group.

### Defining your own resource limits

To set your own resource limits for the number of rows that are retrieved from the database, you can add a resource group to the resource control table.

### About this task

This procedure adds a resource group named GROUP1. The governor prompts users in GROUP1 when the number of rows reaches 10,000 and cancels the user's activity when the number of rows reaches 15,000. For TSO, ISPF, and native z/OS batch, the governor also prompts a user in GROUP1 when processor time reaches 300 seconds, and cancels the user's activity when the processor time reaches 1,000 seconds. The procedure also shows an example of how to add a user to a resource group.

### Procedure

To add a resource group to the resource control table, follow these steps:

1. Set the number of rows at which the user is warned of the approaching resource limit.

If you do not want to warn users when they are approaching their limit for the number of rows, skip to Step "2" on page 286.

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','ROWPROMPT',10000)
```

*Figure 85. Configuring a prompt for 10,000 retrieved rows*

2. Set the number of rows at which the governor cancels the user's activity. The following example sets the cancellation at 15,000 rows.

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','ROWLIMIT',15000)
```

*Figure 86. Configuring cancellation of QMF activity at 15,000 retrieved rows*

3. Set the processor time that elapses before the user is warned of the approaching resource limit. Elapsed processor time applies only to TSO, ISPF, and native z/OS batch environments. If you do not want to warn users when they are approaching their limit for the time that is elapsed, skip to Step “4” on page 287.

---

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','TIMEPROMPT',300)
```

*Figure 87. Configuring a prompt to be issued after 300 seconds (5 minutes) of processor time in TSO, ISPF, and native z/OS batch*

- 
4. Set the amount of processor time that can elapse before the governor cancels the user's activity.

---

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','TIMELIMIT',1000)
```

*Figure 88. Configuring cancellation of QMF activity after 1000 seconds (over 16 minutes) of processor time in TSO, ISPF, and native z/OS batch*

- 
5. Set the real time between intervals when the governor checks the user's activity.

---

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','TIMECHECK',800)
```

*Figure 89. Configuring QMF to compare user activity against resource limits approximately every 13 minutes in TSO, ISPF, and native z/OS batch*

- 
6. Turn on governing for the GROUP1 resource group.

SCOPE is a resource option that activates or deactivates governing. Each resource group in Q.RESOURCE\_TABLE must have a RESOURCE\_OPTION called SCOPE, and SCOPE must have a corresponding INTVAL of zero, or the resource group is not governed. Set INTVAL to 1 to deactivate governing.

---

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','SCOPE',0)
```

*Figure 90. Turning on the governor for the GROUP1 resource group*

- 
7. Add user JONES to the GROUP1 resource group in the English QMF environment.

---

```
UPDATE Q.PROFILES
SET RESOURCE_GROUP='GROUP1'
WHERE CREATOR='JONES' AND
TRANSLATION='ENGLISH'
```

*Figure 91. Updating a user's resource group in the Q.PROFILES table*

---

Use a similar statement to update user profiles in an NLF environment, but use a value for TRANSLATION that represents the name that QMF uses for the NLF. For the name that QMF uses for an NLF, see [Table 27 on page 102](#)

## What to do next

Instruct the user whose profile you updated to end the current QMF session and start another to activate the new values.

### Related tasks

[Modifying the default governor exit routine or writing your own routine](#)

If you decide to govern resources other than the number of rows returned from the database or the processor time expired, you need to modify the default governor exit routine or write your own.

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### Creating your own resource control table

You can create your own table or rename Q.RESOURCE\_TABLE.

### About this task

You can include additional columns in the table you create if Q.RESOURCE\_VIEW is the view defined on this table and the table includes the columns structure of the Q.RESOURCE\_TABLE table.

**Important:** When running QMF for TSO and CICS, you invalidate the QMF application plan when you drop the view. For this reason, you should work outside of QMF when you drop and re-create the resource table and view.

### Procedure

To create your own resource table, follow these steps. These example SQL statements create a table called MY\_RESOURCES. Substitute your own table, column, and table space names in the statement.

1. Erase Q.RESOURCE\_TABLE from the database with a statement like the following, because Q.RESOURCE\_VIEW is defined on this table:

```
DROP TABLE Q.RESOURCE_TABLE
```

2. Re-create the table.

Under TSO, substitute your own table space name for DSQTSSN1.

```
CREATE TABLE MY_RESOURCES
(GROUP_NAME CHAR(16) NOT NULL,
 CONSTRAINT CHAR(16) NOT NULL,
 INTEGER INTEGER,
 FLOAT_VALUE FLOAT,
 CHARACTER VARCHAR(80))
IN DSQTSSN1
```

*Figure 92. Creating a resource control table or renaming Q.RESOURCE\_TABLE*

3. If you do not use the table space supplied with QMF, you must create your own. If you rebind the QMF application plan explicitly, you also need the BIND privilege on the plan.
4. Redefine Q.RESOURCE\_VIEW as a view on the new table, MY\_RESOURCES.

Always re-create Q.RESOURCE\_VIEW if you decide to use a table other than Q.RESOURCE\_TABLE or decide to give Q.RESOURCE\_TABLE a different name, because QMF queries the view, not the table, to obtain resource control information to pass to the governor exit routine.

The following example shows how to redefine Q.RESOURCE\_VIEW as a view on the new table, MY\_RESOURCES. Substitute your own table and column names for those in the example.

```
CREATE VIEW Q.RESOURCE_VIEW
(RESOURCE_GROUP, RESOURCE_OPTION, INTVAL, FLOATVAL, CHARVAL)
```

```
AS SELECT GROUPNAME, CONSTRAINT, INTEGER, FLOAT_VALUE, CHARACTER
FROM MY_RESOURCES
```

5. Grant the SELECT privilege on Q.RESOURCE\_VIEW to PUBLIC.
6. Test the new view; you can test the view using SPUFI. Finally, rebind the QMF application plan.

### Related concepts

How the governor knows what the resource limits are

Each row of the Q.RESOURCE\_TABLE supplied by IBM contains resource control information.

## Program components of the governor exit routine

The names of the member components of the governor exit routine vary by environment and language you installed (English or an NLF).

The following table shows the names of the governor exit routine components and the purpose that each component serves. Replace the *n* character in the names shown with the 1-character language identifier that matches the NLF you are using. In the component names, a 1 represents TSO, ISPF, and native z/OS; 3 represents CICS.

*Table 66. Components of the default governor*

| Environment                       | Member name | Library                  | Function                                                                                                                                                          |
|-----------------------------------|-------------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>TSO, ISPF, and native z/OS</b> | DSQUnGV1    | QMF1210.SDSQLOAD         | Load module for TSO, ISPF, and native z/OS                                                                                                                        |
|                                   | DSQUnGV1    | QMF1210.SDSQUSR <i>n</i> | Source code for the governor exit routine for TSO, ISPF, and native z/OS                                                                                          |
|                                   | DXEUnGV1    | QMF1210.SDSQUSR <i>n</i> | Contains text and related definitions for the governor prompts and cancellation messages for TSO, ISPF, and native z/OS                                           |
| <b>CICS</b>                       | DSQUnGV3    | QMF1210.SDSQLOAD         | Load module for CICS                                                                                                                                              |
|                                   | DSQUnGV3    | QMF1210.SDSQUSR <i>n</i> | Source code for the governor exit routine for CICS                                                                                                                |
|                                   | DXEUnGV3    | QMF1210.SDSQUSR <i>n</i> | Contains text and related definitions for the governor cancellation message in CICS                                                                               |
|                                   | DXEUnGM     | QMF1210.SDSQUSR <i>n</i> | Contains BMS map for the governor prompts in CICS                                                                                                                 |
| <b>All environments</b>           | DXEGOVA     | QMF1210.SDSQUSR <i>n</i> | DSECT for the DXEGOVA control block                                                                                                                               |
|                                   | DXEXCBA     | QMF1210.SDSQUSR <i>n</i> | DSECT for the DXEXCBA control block<br><br>In QMF Version 12.1, this control block has three new fields that provide governing for SQL queries larger than 32 KB. |

If you are using an NLF, you can govern resources in an NLF session as well as an English QMF session, by using different versions of the module DSQUnGVx for each language environment. For example, if you have both English and German installed, use the module DSQUEGV1 for English in TSO, ISPF, and native z/OS batch and the module DSQUDGV1 for German in TSO, ISPF, and native z/OS batch.

You can share the resource control table (Q.RESOURCE\_TABLE or one you create yourself) and the Q.RESOURCE\_VIEW between language environments, just as the Q.PROFILES table can contain profiles for English or any NLF.

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## How QMF and the governor interact

QMF interacts with the governor exit routine differently in different environments.

At the start of a user's session, QMF brings the governor into the user's virtual storage. Under TSO, ISPF, and native z/OS, QMF issues a LOAD command. Under CICS, QMF issues an EXEC CICS LOAD command to bring the governor into the user's virtual storage.

The load module library QMF1210.SDSQLOAD is assumed to be in a library concatenated to the user's STEPLIB data set.

For performance reasons, an Assembler call interface is used between QMF and the governor exit routine. The governor exit routine must provide fast performance because, depending on which resources you are trying to control, it might be called on every row retrieved from the database.

This figure shows the program structure of a governor exit routine under TSO, ISPF, or native z/OS.

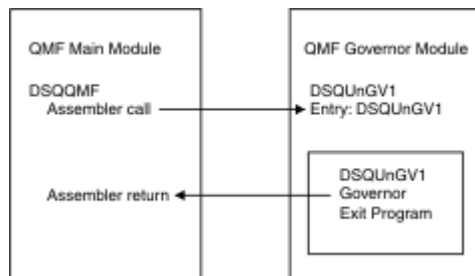


Figure 93. How QMF interacts with the governor exit under TSO, ISPF, or native z/OS

The CICS control block interface to the governor exit consists of the following parts:

- Interface control blocks DXEXCBA and DXEGOVA, which are shipped with QMF
- Prolog and epilog macros DFHEIENT and DFHEIRET, which are shipped with CICS
- Command interface modules DFHEAI and DFHEAI0, which are shipped with CICS
- The governor exit program, which is named DSQUnGV3



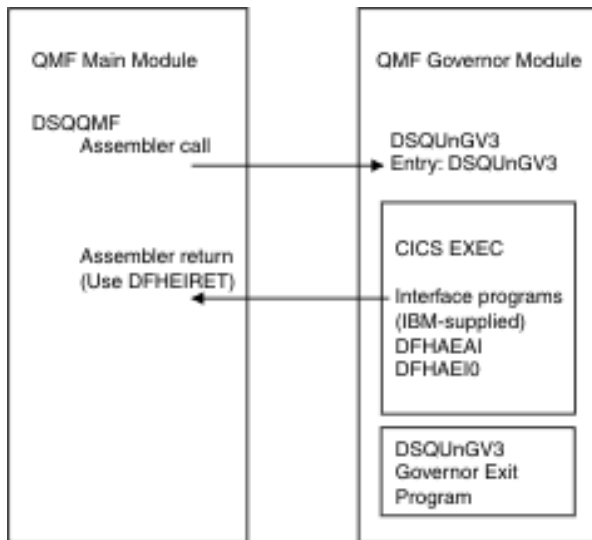


Figure 94. How QMF interacts with the governor exit in a CICS-supplied environment

The governor exit routine executes on the same program level as the main QMF program.

Under CICS, the entry point to the governor exit routine is DSQUnGV3. When it calls the governor exit routine, QMF always branches to the address returned by CICS as the result of an EXEC CICSLOAD command.

If the load fails or the module does not support 31-bit addressing mode, QMF logs a warning message, disables the governor exit, and continues the session without the governor.

### How and when QMF calls the governor exit routine

When you create or modify a governor exit routine, you must establish addressability to the exit routine for the points at which QMF calls the routine.

#### Points at which QMF calls the governor

Function calls to the governor exit routine either precede or follow a specific type of QMF activity.

The following figure provides an overview of how the governor limits use of resources. QMF issues standard Assembler CALL statements to the governor exit routine. The term *function calls* describes the points during the QMF session when these CALL statements are issued.

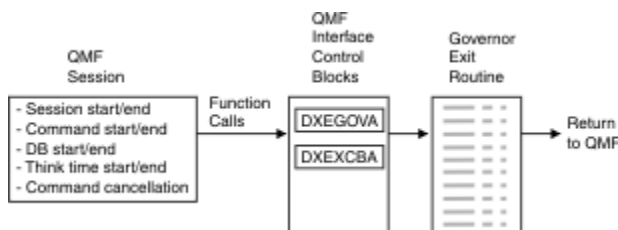


Figure 95. How a governor exit routine works with QMF for TSO and CICS

Function calls are made at the following points:

Table 67. Points at which QMF makes function calls to the governor exit routine. Function calls to the governor exit routine either precede or follow a specific type of QMF activity.

| QMF activity                                          | Description of the function call to the governor exit routine                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>At the beginning and end of a QMF session</b>      | QMF calls the governor exit routine during initialization for a QMF session, after the governor exit routine is loaded into the user's virtual storage. The governor initializes itself for the session using the resource control information contained in rows passed from the query of Q.RESOURCE_VIEW.                                                                                                                                                                                                                                                                                                            |
| <b>After a new connection is made to the database</b> | <p>When a user issues the CONNECT command, the Q.PROFILES table and the resource control table are re-initialized. The governor is called because the resource control values might have changed if a different CONNECT ID was used. All unfinished database operations are completed before the connection is made.</p> <p>Although the governor exit routine cannot cancel a connection to the database, you can write statements in your own routine that cancel the user's session on the next activity if the resource information passed to the governor indicates that the user is not allowed to use QMF.</p> |
| <b>Before and after running a command</b>             | QMF calls the governor before and after running all commands. There can be several calls for the start of commands before a call for the completion of a command. For example, a RUN PROC command results in two "start command" calls and two "end command" calls when there is a RUN QUERY command embedded in the procedure.                                                                                                                                                                                                                                                                                       |

Table 67. Points at which QMF makes function calls to the governor exit routine. Function calls to the governor exit routine either precede or follow a specific type of QMF activity. (continued)

| QMF activity                                                   | Description of the function call to the governor exit routine                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Before database activity starts and when it ends</b></p> | <p>QMF calls the governor just before it begins a variety of database operations, such as PREPARE, OPEN, and FETCH; QMF also calls the governor upon completing any database activity.</p> <p>When QMF retrieves data, it fits the maximum number of rows possible into a buffer that has a minimum size of 4 KB. QMF calls the governor once upon retrieving the first row into the buffer and once upon either filling the buffer or reaching the end of the table, whichever comes first.</p> <p>QMF also calls the governor when SQL, QBE, or prompted queries are submitted using a RUN QUERY command, or when QMF is running SQL statements required by a QMF command. For example, a SAVE DATA command might result in DELETE, CREATE, and INSERT statements. The governor is called before and after each of these operations. If there is an incomplete DATA object when a command is entered, there might be governor calls for database activity while the DATA object is being completed.</p> <p>The following QMF commands always force database activity:</p> <ul style="list-style-type: none"> <li>• DISPLAY (when the object is a table or view)</li> <li>• EDIT TABLE</li> <li>• ERASE TABLE</li> <li>• EXPORT TABLE</li> <li>• IMPORT TABLE</li> <li>• PRINT (when the object is a table or view)</li> <li>• RUN (for queries)</li> <li>• SAVE DATA (which results in one or more implicit CREATE TABLE statements)</li> <li>• Scrolling commands that result in fetching data when a report is being displayed</li> <li>• Data retrieval operations (fetch operations)</li> </ul> |

Table 67. Points at which QMF makes function calls to the governor exit routine. Function calls to the governor exit routine either precede or follow a specific type of QMF activity. (continued)

| QMF activity                                           | Description of the function call to the governor exit routine                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Before and after the user makes a choice</b></p> | <p>At various points in a session, QMF waits for users to make decisions. The time QMF spends waiting is known as <i>think time</i>.</p> <p>QMF calls the governor before performing an operation that leads to think time, such as displaying a panel for a user-entered selection. As soon as the user enters a response and ends the period of think time, QMF calls the governor.</p> <p>Any of the following activities lead to think time:</p> <ul style="list-style-type: none"> <li>• Displaying a QMF panel between running commands</li> <li>• Displaying help panels</li> <li>• Displaying confirmation prompt panels; for example, when the user is about to erase something by issuing the SAVE command that replaces the object</li> <li>• Displaying command prompt panels; for example, when the user enters DISPLAY ?</li> <li>• Displaying the LIST prompt panel</li> <li>• Displaying ICU and EXTRACT panels</li> <li>• Running the EDIT PROC and EDIT QUERY functions</li> </ul> |
| <p><b>At initiation of an abnormal ending</b></p>      | <p>QMF calls the governor just before it initiates an abnormal ending. The governor can perform the cleanup necessary before abend processing begins. The actions might be similar to those during the session end.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### Related concepts

#### Managing QMF performance

You can monitor the performance of QMF and find some solutions or workarounds to common performance issues.

#### What happens upon entry to the governor exit routine

QMF calls the governor exit routine by branching to the address of entry point DSQUnGV1 (TSO, ISPF, or native z/OS) or DSQUnGV3 (CICS).

When it calls the governor, QMF always branches to an entry point named DSQUnGV1 (for TSO, ISPF, or native z/OS) or DSQUnGV3 (for CICS), where *n* is a 1-character national language identifier. For this reason, you cannot use the entry point to determine the type of exit; instead, use the GOVFUNCT control-block field. Its value is a positive integer that identifies the type of exit. Each type of function call has a specific value for the GOVFUNCT field. These values are shown in [Figure 99 on page 296](#).

#### Branching to the DSQUnGV1 entry point in TSO, ISPF, or native z/OS

QMF calls the governor exit routine in TSO, ISPF, or native z/OS by branching to the address of entry point DSQUnGV1. Upon entry to the governor exit routine:

- Register 1 contains the address of the parameter list.

The parameter list contains two full-word addresses: one for the DXEXCBA control block and the other for the DXEGOVA control block.

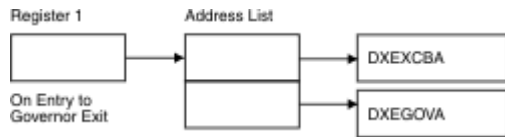


Figure 96. Contents of register 1 on a call to the governor exit routine in TSO, ISPF, or native z/OS

- Register 13 contains the address of the QMF SAVE area.
- Register 14 contains the return address from the call.
- Register 15 contains the address of the entry point, which is DSQUEGV1.

After the governor is called, it might begin with code like this code sample, which is from the default governor for TSO, ISPF, or native z/OS.

```

DSQUEGV1 CSECT
          USING *,R15
          B    FENTRY          BRANCH AROUND CONSTANTS
          DC   C'DSQUEGV1'     MODULE NAME
          DC   C' '
          DC   C'&SYSDATE '    DATE OF ASSEMBLY
          DC   C'&SYSTEME '    TIME OF ASSEMBLY
          DS   0H

*
FENTRY   STM   R14,R12,12(R13)  SAVE THE REGISTERS
          BALR  R12,0           INITIALIZE BASE REGISTER
          DROP R15
          LA   R02,MAINSV      CHAIN THE SAVE AREAS
          ST   R02,8(R13)
          ST   R13,MAINSV+4
          LR   R13,R02

*
          L    R01,4(R01)       GET ADDRESS OF DFHCOMMA
          L    XCBPTR,0(R01)    GET ADDRESS OF QMF EXIT CTL BLK
          L    GOVPTR,4(R01)    GET ADDRESS OF QMF GOV CTL BLK
          USING DXEXCBA,XCBPTR
          USING DXEGOVA,GOVPTR
          LA   WORKPTR,GOVUSERS SCRATCH PAD ADDRESS
          USING WORK,WORKPTR

:
MAINSV   DS    18F            SAVE AREA
XCBPTR   EQU   R02           PTR TO DXEXCBA CONTROL BLOCK
GOVPTR   EQU   R03           PTR TO DXEGOVA CONTROL BLOCK
WORKPTR  EQU   R04           PTR TO SCRATCH_PAD AREA
  
```

Figure 97. Sample code that shows the start of a governor session in TSO, ISPF, or native z/OS

The code in the preceding sample first branches around a block of constants that can serve as eye catchers in a dump of virtual storage. The constants name the entry point and the applicable version of QMF. They also show the date and time that the code was assembled.

The code establishes base registers for the program, pointers to the DXEXCBA and DXEGOVA control blocks, and a scratchpad area named GOVUSERS. The scratchpad area is preserved by QMF between calls to the governor. A DSECT named WORK describes this scratchpad area in the code for the default governor.

After the governor processes a call, it returns control to QMF in the standard way. You must use the standard epilog and prolog, as in the sample code for the default governor that is shown here:

```

L    R13,4(R13)      RESTORE CALLER'S SAVE AREA ADDRESS
      LM   R14,R12,12(R13)  RESTORE CALLER'S REGISTERS
      XR   R15,R15        ZERO RETURN CODE
      BR   R14           RETURN TO CALLER
  
```

Figure 98. How the default governor exit routine returns control to QMF in TSO, ISPF, or native z/OS

## Branching to the DSQUnGV3 entry point in CICS

Entry to the governor exit routine in CICS follows the standard CICS linkage conventions:

- Register 1 contains a CICS parameter list suitable for processing by the macros DFHEIENT and DFHEIRET that are supplied with CICS. The following figure shows the contents of register 1 on a call to the governor.

DFHEIBLK is the address of the CICS communications area. DFHCOMMA contains two pointers, one to the DXEXCBA control block and the other to the DXEGOVA control block.

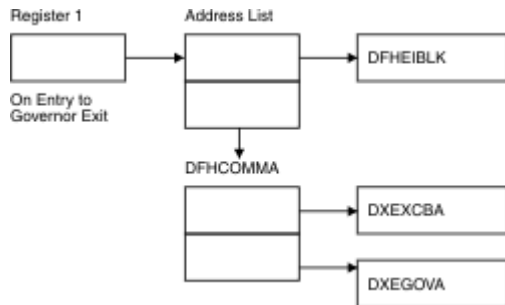


Figure 99. Contents of register 1 on a call to the governor exit routine in CICS

- Register 13 contains the address of a standard CICS working storage area as described by CICS DSECT DFHEISTG.
- Register 14 contains the return address.

Because the governor program runs on the same program level as QMF, use caution when you use any EXEC CICS commands that change the environment (for example, CICS HANDLE CONDITION). If you need to use the CICS HANDLE CONDITION statement, use EXEC CICS PUSH and EXEC CICS POP statements to save and restore the existing handle conditions.

Begin the governor program with code similar to the following.

```

DSQUEGV3 TITLE 'QMF GOVERNOR EXIT ROUTINE'
DFHEISTG DSECT
DSQUEGV3 DFHEIENT CODEREG=(12),DATAREG=(13),EIBREG=(10)
          B          FENTRY          BRANCH AROUND CONSTANTS
*
MODNAME  DC      C'DSQUEGV3'          MODULE NAME
          DC      C' '
          DC      C'&SYSDATE '        DATE OF ASSEMBLY
          DC      C'&SYSTEMTIME '     TIME OF ASSEMBLY
          DS      0H
*
FENTRY   DS      0H
          L       R01,4(R01)          GET ADDRESS OF DFHCOMMA
          L       XCBPTR,8(R01)       GET ADDRESS OF QMF EXIT CTL BLK
          L       GOVPTR,12(R01)     GET ADDRESS OF QMF GOV CTL BLK
          USING   DXEXCBA,XCBPTR
          USING   DXEGOVA,GOVPTR
          LA      WORKPTR,GOVUSERS    GET ADDRESS OF GOVERNOR WORK AREA
          USING   WORK,WORKPTR
*
          :
          GOVPTR EQU R03              PTR TO DXEGOV CONTROL BLOCK
          XCBPTR EQU R02              PTR TO DXEXCB CONTROL BLOCK
          WORKPTR EQU R04             PTR TO GOVERNOR SCRATCH PAD AREA
  
```

Figure 100. Sample code at the start of a governor session (for CICS)

The code in the preceding sample first branches around a block of constants that can serve as eye catchers in a dump of virtual storage. The constants name the entry point and the applicable version of QMF. They also show the date and time that the code was assembled.

The code establishes base registers for the program, pointers to the DXEXCBA and DXEGOVA control blocks, and a scratchpad area named GOVUSERS. The scratchpad area is preserved by QMF between calls to the governor. A DSECT named WORK describes this scratchpad area in the code for the default governor.

When processing is complete, the governor returns control to QMF with the standard CICS return as specified by the CICS macro DFHEIRET.



**Attention:** Do not use the command EXEC CICS RETURN. This command ends the QMF session without releasing QMF resources.

The governor program ends with code similar to the following.

```

:
*
      XR      R15,R15          ZERO RETURN CODE
      DFHEIRET RCREG=15
*

```

Figure 101. Ending code for the governor program in CICS

### Related concepts

Structure of the DXEGOVA control block

The DXEGOVA control block passes to the governor exit routine information about a user's resource constraints. This information is located in a resource control view called Q.RESOURCE\_VIEW.

### Related tasks

Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### Establishing addressability for function calls

Because QMF always branches to an entry point named DSQUnGV1 (TSO, ISPF, or native z/OS) or DSQUnGV3 (CICS) when it calls the governor, you cannot use these entry points to determine the type of function call; instead, use the GOVFNCT field of the DXEGOVA control block.

In the IBM-supplied governor exit routine, GOVFNCT contains a character value that identifies the type of function call. This character value, in turn, equates to a 1-byte binary integer from 1 to 10. For example, on a function call for the start of a QMF session, the value of GOVFNCT is GOVINIT, which equates to a numeric value of X'1'.

Both character and numeric values for each type of function call are shown in this figure. GOVABEND is not called when running in CICS.

|          |     |    |       |                           |
|----------|-----|----|-------|---------------------------|
| GOVINIT  | EQU | 1  | ----- | INITIALIZATION OF SESSION |
| GOVTERM  | EQU | 2  | ----- | TERMINATION OF SESSION    |
| GOVSCMD  | EQU | 3  | ----- | START COMMAND             |
| GOVECMD  | EQU | 4  | ----- | END COMMAND               |
| GOVCONN  | EQU | 5  | ----- | CONNECT COMMAND           |
| GOVSDBAS | EQU | 6  | ----- | START DATABASE            |
| GOVEDBAS | EQU | 7  | ----- | END DATABASE              |
| GOVSACTV | EQU | 8  | ----- | SUSPEND QMF ACTIVITY      |
| GOVRACTV | EQU | 9  | ----- | RESUME QMF ACTIVITY       |
| GOVABEND | EQU | 10 | ----- | QMF ABEND OPERATION       |

Figure 102. Character and numeric values for the GOVFNCT field of the DXEGOVA control block

To improve performance in your own exit routine, you can follow the convention the IBM-supplied governor uses, and equate the values of GOVFNCT with binary numbers by using a branch table. QMF uses the branch table to find the addresses to branch to for each type of function call.

The following example code identifies branch addresses for the default governor.

```

XR      R07,R07          ZERO REGISTER 7
        IC      R07,GOVFUNCT      IDENTIFY EXIT TYPE
        SLL     R07,2             DETERMINE BRANCH TABLE OFFSET
        LA      R15,FUNBTAB(R07)  GET BRANCH TABLE ADDRESS
        L       R15,0(R15)       GET BRANCHING ADDRESS
        BALR    R14,R15          BRANCH TO THE APPROPRIATE CODE
        .
        .
        .
        .
        .
        .
FUNBTAB DS      0F
        DC      A(BYPASS)        VALUE "0" - UNUSED
        DC      A(INIT)          VALUE "1" - QMF INITIALIZATION
        .
        .
        .
        DC      A(SUSPEND)       VALUE "10" - QMF ABEND IN PROCESS
    
```

Figure 103. Identifying the type of function call and branching to the appropriate address

**Modifying the default governor exit routine or writing your own routine**

If you decide to govern resources other than the number of rows returned from the database or the processor time expired, you need to modify the default governor exit routine or write your own.

**Before you begin**

- Read [“Program components of the governor exit routine”](#) on page 289.
- Read [“How QMF and the governor interact”](#) on page 290.
- Establish addressability to the exit routine for the points at which QMF calls the routine. [“How and when QMF calls the governor exit routine”](#) on page 291 explains this step.

**About this task**

One of the control blocks that is used to pass resource control information to the governor exit, DXEXCBA, has been modified in QMF Version 12.1 to accommodate governing of SQL queries that are larger than 32 KB. If you plan to take advantage of this feature and you modified the default governor in prior QMF releases or you have a user-written governor routine already in place, you need to modify the routine to make use of the new control block fields that provide this feature.

**Related concepts**

Structure of the DXEXCBA control block

The DXEXCBA control block passes to the governor exit routine information about the state of the QMF session upon entry to the governor. The governor combines this information with information about resource limits (contained in DXEGOVA) to determine when the resource limits have been exceeded and when to cancel the user's activity.

**Passing resource control information to the governor exit**

QMF passes resource control information using two control blocks named DXEGOVA and DXEXCBA. Their addresses are passed to the governor on every function call.

Before you begin, you should know how to define your own resource limits in a format that the governor can use.

The DSECT DXEXCBA (shipped as DXEXCBA) and the DSECT DXEGOVA (shipped as DXEGOVA) are located in QMF1210.SDSQSURE. Include these DSECTs in your program using the Assembler COPY statement.

Three new fields have been added to the DXEXCBA control block in QMF Version 12.1 to provide governing of SQL queries that are larger than 32 KB: XCBQRYPT, XCBQRYP2, and XCBQRYL2. If you



modified the default governor exit routine in an earlier release or wrote your own exit, some migration steps are required to be compatible with this release and to take advantage of this feature.

### Related concepts

[Forward compatibility of earlier releases with QMF Version 12 Release 1](#)

Most objects that were created under earlier releases can be used in QMF Version 12.1 without modification.

### Related tasks

[Defining your own resource limits](#)

To set your own resource limits for the number of rows that are retrieved from the database, you can add a resource group to the resource control table.

### Structure of the DXEGOVA control block

The DXEGOVA control block passes to the governor exit routine information about a user's resource constraints. This information is located in a resource control view called Q.RESOURCE\_VIEW.

The following table provides the name of each field in the DXEGOVA control block, with its data type and purpose. Each data type is listed as it appears in the DS statement that defines the field in the DSECT. For example, for the GOVOROWS field, the letter F indicates that this field contains a full-word integer. The DS statement for GOVOROWS appears as GOVOROWS DS F.

The layout of the control blocks and the information that they contain applies to all QMF environments. Therefore, some of the information shown in the control blocks might not apply to QMF in the environment that you are using.

| Field    | Data type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GOVCADDR | A         | Contains the address to branch to for canceling an activity.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| GOVFUNCT | XL1       | Indicates the type of function call. Possible values are: <ul style="list-style-type: none"> <li>• GOVINIT (session initialization); GOVTERM (session termination)</li> <li>• GOVSCMD (start command); GOVECMD (end command)</li> <li>• GOVCONN (connect command)</li> <li>• GOVSDBAS (start database retrieval operation); GOVEDBAS (end database retrieval operation)</li> <li>• GOVSACTV (suspend QMF activity for user think time); GOVRACTV (resume QMF activity)</li> <li>• GOVABEND (start of an abnormal ending)</li> </ul> |
| GOVGROUP | CL16      | Contains the name of the user's resource group. This value does not change during a QMF session.                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| GOVNAME  | CL8       | Contains the name of the control block (DXEGOVA). This value does not change during a session. It can serve as an eye catcher in a dump of virtual storage.                                                                                                                                                                                                                                                                                                                                                                         |
| GOVOROWS | F         | Contains the number of rows for the user's resource group in the resource control table. This value does not change during a session, and can be zero.                                                                                                                                                                                                                                                                                                                                                                              |

Table 68. Fields of the DXEGOVA interface control block to the governor (continued)

| Field    | Data type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GOVRESC  | 10XL128   | <p>Contains information from the resource control table. This information is divided into 10 contiguous blocks of storage that are structured like DSECT GOVRESCT. A block contains information about one of the rows for the user's resource group in the QMF resource control table.</p> <ul style="list-style-type: none"> <li>• If the resource group has less than 10 rows, unused blocks are those at the end of the field.</li> <li>• If the resource group has more than 10 rows, use the GOVNEXTR field (in the GOVRESCT DSECT) to access additional rows.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| GOVRESCT | DSECT     | <p>Describes the block of storage containing information about one of the user's rows of the resource control table.</p> <p><b>GOVOPTN(CL16)</b><br/>Contains the value in the RESOURCE_OPTION column of the resource control table. Blocks in the chain are ordered alphabetically on the content of this field.</p> <p><b>GOVNULLI(H)</b><br/>Null indicator for INTVAL column.</p> <p><b>GOVINTVL(F)</b><br/>Value of INTVAL column.</p> <p><b>GOVNULLF(H)</b><br/>Null indicator for FLOATVAL column.</p> <p><b>GOVFLOAT(D)</b><br/>Value of FLOATVAL column.</p> <p><b>GOVNULLC(H)</b><br/>Null indicator for CHARVAL column.</p> <p><b>GOVCHLEN(H)</b><br/>Length of data in CHARVAL column.</p> <p><b>GOVCHAR(CL80)</b><br/>Value in CHARVAL column.</p> <p><b>GOVNEXTR(A)</b><br/>Points to the block of data for the next resource table row. Contains zero if this is the last row.</p> <p>Any null indicator in the structure is zero when its corresponding column value isn't null. If the column value is null, the indicator is not zero.</p> |
| GOVSQLCA | A         | Address of the SQL communications area (SQLCA), which holds information about the SQL SELECT query on the resource control view (Q.RESOURCE_VIEW).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| GOVSQLRC | F         | Return code from the SQL SELECT query on the resource control view (Q.RESOURCE_VIEW). If it is nonzero, the query failed and no rows are passed to the governor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| GOVUSERS | CL2048    | Scratchpad area, retained between session calls. QMF does not change this value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

The following figure shows the structure of the DXEGOVA control block.

```

***** 00001000
*      * 00002000
*      * 00003000
*      * 00004000
CONTROL BLOCK NAME: DXEGOVA

```

```

*          FUNCTION:
*
*          THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND
*          THE GOVERNOR EXIT ROUTINE.
*
*          STATUS: Version 12.1 RELEASE 1 LEVEL 0
*
*          INNER CONTROL BLOCKS: NONE
*
*          CHANGE ACTIVITY: NA
*
*          CHANGE DATE: NA
*
*****
*
DXEGOVA  DSECT
          DS      0D
GOVNAME  DS      CL8      -- CONTROL BLOCK IDENTIFICATION
          SPACE
GOVEXCTL DS      XL72     -- EXIT CONTROL
          ORG    GOVEXCTL
GOVFUNCT DS      XL1     ----- FUNCTION CODE
GOVINIT  EQU     1       ----- INITIALIZATION OF SESSION
GOVTERM  EQU     2       ----- TERMINATION OF SESSION
GOVSCMD  EQU     3       ----- START COMMAND
GOVECMD  EQU     4       ----- END COMMAND
GOVCONN  EQU     5       ----- CONNECT COMMAND
GOVSDBAS EQU     6       ----- START DATA BASE
GOVEDBAS EQU     7       ----- END DATA BASE
GOVSACTV EQU     8       ----- SUSPEND QMF ACTIVITY
GOVRACTV EQU     9       ----- RESUME QMF ACTIVITY
GOVABEND EQU    10      ----- QMF ABEND OPERATION
GOVPAD10 DS      CL7     ----- RESERVED FIELD
          SPACE
GOVCADDR DS      A      ---- ADDR TO BRANCH TO FOR CANCELLATION
          SPACE
GOVOROWS DS      F      ---- NUMBER OF OPTION ROWS RETRIEVED
          SPACE
GOVSQLRC DS      F      ---- RESOURCE TABLE SQL RETURN CODE
          SPACE
GOVSQLCA DS      A      ---- ADDRESS OF SQLCA FOR ERROR CONDITION
          SPACE
GOVGROUP DS      CL16    ---- GROUP NAME
GOVPAD20 DS      CL32    ---- RESERVED FIELD
          SPACE
GOVUCTL  DS      XL304   -- USER CONTROL AREA
          ORG    GOVUCTL
GOVUSERS DS      CL2048  ---- USER SCRATCH PAD AREA
GOVPAD30 DS      CL48    ---- RESERVED FIELD
          SPACE
          DS      0D
GOVRESC  DS      10XL128 -- RESOURCE CONTROL TABLE
          ORG    GOVRESC
GOVRESCT DSECT
          DS      0D
          DS      CL16    ----- RESOURCE OPTION
GOVNULLI DS      H      ----- INTEGER NULL INDICATOR
GOVPAD40 DS      CL2     ----- RESERVED FIELD
GOVINTVL DS      F      ----- INTEGER OPTION REPRESENTATION
GOVNULLF DS      H      ----- FLOATING POINT NULL INDICATOR
GOVPAD50 DS      CL6     ----- RESERVED FIELD
GOVFLOAT DS      D      ----- FLOATING POINT OPTION REPRESENTATION
GOVNULLC DS      H      ----- CHARACTER NULL INDICATOR
GOVCHLEN DS      H      ----- LENGTH OF THE CHARACTER OPTION
GOVCHAR  DS      CL80   ----- CHARACTER OPTION REPRESENTATION
GOVNEXTR DS      A      ----- POINTER TO NEXT RESOURCE CONTROL ROW

```

Figure 104. The DXEGOVA control block

### Related concepts

#### Addressing the resource control table

The GOVGROUP field of the DXEGOVA control block holds the value of the RESOURCE\_GROUP column of Q.RESOURCE\_VIEW, the view defined on the resource control table.

#### Structure of the DXEXCBA control block

The DXEXCBA control block passes to the governor exit routine information about the state of the QMF session upon entry to the governor. The governor combines this information with information about

resource limits (contained in DXEGOVA) to determine when the resource limits have been exceeded and when to cancel the user's activity.

### Addressing the resource control table

The GOVGROUP field of the DXEGOVA control block holds the value of the RESOURCE\_GROUP column of Q.RESOURCE\_VIEW, the view defined on the resource control table.

All information about the user's resource options is stored in blocks; there is one block for each of the user's resource options that you decide to monitor.

The first block defines the first resource option and is stored in the DXEGOVA control block as the DSECT GOVRESCT. The address of this DSECT is defined in the GOVRESC field of the DXEGOVA control block. You can establish addressability to the GOVRESC field in your own routine using the address of the GOVRESCT DSECT.

Negative half-word integers in the DSECT represent null values entered for INTVAL, CHARVAL, or FLOATVAL in Q.RESOURCE\_VIEW; zero or positive half-words indicate a value in that column of Q.RESOURCE\_VIEW.

The blocks that store the resource control information form a chain in which a pointer in one block points to the beginning of the next block (the next resource option) in the chain. For example, the GOVNEXTR DS statement in the GOVRESCT DSECT contains the address of the next block in the chain of resource control information. Each block in the chain has a GOVNEXTR DS statement. In the final block, the GOVNEXTR DS statement contains zeros to mark the end of the user's resource control information.

The following section of code for the default governor processes the blocks of resource control information. In this code, GOVRESC points to the GOVRESCT DSECT.

```

L      R08,GOVOROWS      GET NUMBER OF RESOURCE TABLE ROWS
      LTR R08,R08        ANY RESOURCE TABLE ROWS?
      BZ  ENDRESST      NO, SKIP RESOURCE INITIALIZATION
      LA  R05,GOVRESC   GET ADDRESS OF 1ST RESOURCE ROW
      USING GOVRESC,R05  BASE RESOURCE RECORD ENTRY
LOOK4RES DS 0H          MAIN LOOP THRU RESOURCE ROWS
      LTR R05,R05      ANY MORE RESOURCE TABLE ROWS?
      BZ  ENDRESST      NO, END RESOURCE INITIALIZATION
      :
      L   R05,GOVNEXTR  GET ADDRESS ON NEXT RESOURCE ROW
      B   LOOK4RES      BEGIN NEXT ITERATION
ENDRESST DS 0H         -- BRANCH HERE WHEN FINISHED READING ALL ROWS

      . . .
      . . .
      . . .
      . . .
DXEGOVA DSECT
      . . .
      . . .
      . . .
GOVRESC DS 10XL128     -- RESOURCE CONTROL TABLE
      ORG GOVRESC
GOVRESCT DSECT        -- DSECT FOR RESOURCE ROW
      . . .
      . . .
GOVNEXTR DS A         -- POINTER TO NEXT RESOURCE ROW
      . . .
      . . .
      . . .

```

Figure 105. Resource initialization

### Related concepts

[Structure of the DXEGOVA control block](#)

The DXEGOVA control block passes to the governor exit routine information about a user's resource constraints. This information is located in a resource control view called Q.RESOURCE\_VIEW.

Structure of the DXEXCBA control block

The DXEXCBA control block passes to the governor exit routine information about the state of the QMF session upon entry to the governor. The governor combines this information with information about resource limits (contained in DXEGOVA) to determine when the resource limits have been exceeded and when to cancel the user's activity.

**Structure of the DXEXCBA control block**

The DXEXCBA control block passes to the governor exit routine information about the state of the QMF session upon entry to the governor. The governor combines this information with information about resource limits (contained in DXEGOVA) to determine when the resource limits have been exceeded and when to cancel the user's activity.

For example, you can define a resource option that does not allow user JONES to issue INSERT or UPDATE statements. You can then write your governor exit routine so that, if the XCBQRYP or XCBQRYP2 field of the DXEXCBA control block indicates an INSERT or UPDATE statement, the governor exit calls the QMF cancellation service to cancel the command.

Three new fields have been added to the DXEXCBA control block in QMF Version 12.1 to provide governing of SQL queries that are larger than 32 KB: XCBQRYPT, XCBQRYP2, and XCBQRYL2. If you modified the default governor exit routine in an earlier release or wrote your own exit, some migration steps are required to be compatible with this release and to take advantage of this feature.

The following table provides the name of each field in the control block, with its data type and purpose. Each data type is listed as it appears in the DS statement that defines the field in the DSECT.

The layout of the control blocks and the information they contain is the same for QMF support in all operating environments. Therefore, some of the information shown in the control blocks might not apply to the environment that you are using.

| <i>Table 69. Fields of the DXEXCBA interface control block to the governor</i> |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------------------------------------------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Field</b>                                                                   | <b>Data type</b> | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| XCBACTIV                                                                       | CL1              | <p>Indicates the current type of database activity. Applies only when rows are being retrieved for the current DATA object. Does not apply when rows are retrieved for an IMPORT command. Possible values are:</p> <ol style="list-style-type: none"> <li><b>1</b> OPEN being run</li> <li><b>2</b> FETCH being run</li> <li><b>3</b> PREPARE being run</li> <li><b>4</b> DESCRIBE being run</li> <li><b>5</b> CLOSE being run</li> </ol> <p>This field changes whenever the type of database activity changes. You can use the value when the governor receives control asynchronously as the result of a timer.</p> |

Table 69. Fields of the DXEXCBA interface control block to the governor (continued)

| Field               | Data type | Description                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XCBAIACT            | CL1       | Tells whether the current command is running interactively:<br><b>1</b><br>Interactive<br><b>0</b><br>Noninteractive (batch)<br>Interactive commands display prompt and status panels. This field changes value on any function call for the start of the command; it is reset to zero when the command completes.                                                       |
| XCBAUTH             | CL8       | Contains the user's SQL authorization ID. If the SQL authorization ID is longer than 8 characters, the value is truncated and placed in this field. See XCBAUTHX for the complete SQL authorization ID.                                                                                                                                                                  |
| XCBAUTHX            | CL128     | Contains the user's SQL authorization ID.                                                                                                                                                                                                                                                                                                                                |
| XCBCAN              | CL1       | Indicates that cancellation of the current command was requested (either by the user or the governor). The field is set to 1 if cancellation is requested. Zero indicates that no cancellation was requested. The value changes at the point at which the cancellation is requested. This field is reset to zero before the function call for the command's termination. |
| XCBCLOC             | CL18      | Contains the current location name.                                                                                                                                                                                                                                                                                                                                      |
| XCBCMDL             | F         | Contains the length of the string containing the command to be run. This is the string addressed by the XCBCMDP field. This field changes in value when XCBCMDL changes values.                                                                                                                                                                                          |
| XCBCMDP             | A         | Points to the string containing the command to be run. This field is reset when QMF validates a command at some point before the function call for the start of the command.<br><br>The field is reset to zeros before the function call when the command completes. If a command synonym is being run, it appears here.                                                 |
| XCBCVERB            | CL18      | Holds the verb of the current command. This field changes value on the function call for the start of a command. The value does not change between calls.                                                                                                                                                                                                                |
| XCDBBMG             | CL1       | Identifies the database manager. This value is set to 2 for Db2 for z/OS.                                                                                                                                                                                                                                                                                                |
| XCBE MODE           | CL1       | Indicates the current mode of the QMF session:<br><b>1</b><br>Interactive<br><b>2</b><br>Noninteractive (batch or server)<br>This value does not change during a session.                                                                                                                                                                                                |
| XCBERRET            | F         | Contains the return code to be used in the default cancellation message.                                                                                                                                                                                                                                                                                                 |
| XCBINCI (ISPF only) | CL1       | Tells whether the current command is being run through the command interface. The field is set to 1 if it is, and 2 if it isn't.                                                                                                                                                                                                                                         |
| XCBINPRC            | CL1       | Tells the governor where a command is being run: 1 indicates it is running in a procedure or LIST command; 0 indicates it is being run another way.                                                                                                                                                                                                                      |

Table 69. Fields of the DXEXCBA interface control block to the governor (continued)

| Field                   | Data type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XCBKPARAM               | CL1       | Tells the governor how the DSQSDBCS program parameter is set. The value does not change during a session. Possible values are: 0 for Latin letters; 1 for double-byte character set (DBCS) data.                                                                                                                                                                                                                                                                                                                             |
| XCBLOGM                 | CL1       | Indicates whether QMF should log a message in the QMF trace data set. Use a value of 1 to log the message, and 0 to not log the message.                                                                                                                                                                                                                                                                                                                                                                                     |
| XCBMGTX                 | CL78      | Contains the text for a message. The message can be logged in the QMF trace data, displayed on the screen, or both.                                                                                                                                                                                                                                                                                                                                                                                                          |
| XCBMSGNO<br>(ISPF only) | CL8       | Contains the message ID for an ISPF message definition that can be used to log a message in the trace data, display a message on the screen, or both.                                                                                                                                                                                                                                                                                                                                                                        |
| XCBNAME                 | CL8       | Contains the control block name (DXEXCBA). Can serve as an eye catcher in a dump of virtual storage. This value does not change during a session.                                                                                                                                                                                                                                                                                                                                                                            |
| XCBNLANG                | CL1       | Identifies NLFs being used. This value does not change during a session.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| XCBPANEL (ISPF<br>only) | CL8       | Contains the panel ID for the message help panel for a cancellation message.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| XCBPLAN                 | CL8       | Contains the application plan ID for QMF. This value does not change during a session. This field does not apply in CICS.                                                                                                                                                                                                                                                                                                                                                                                                    |
| XCBQCE                  | F         | Contains the decimal equivalent of the value of the SQLDERRD(4) field in the SQLCA returned from the database. The integer part of this decimal appears on the Database Status panel as a "relative cost" estimate. You can prevent this cost estimate from displaying on the panel by setting the DSQDC_COST_EST global variable to 0.<br><br>The value of the XCBQCE field is set to zero on the function call when the command finishes running. The field contains zeros if the operation is not a data retrieval query. |
| XCBQERR                 | CL1       | Tells whether a QMF error occurred since the previous function call: 0 indicates no error occurred; 1 indicates an error occurred.                                                                                                                                                                                                                                                                                                                                                                                           |
| XCBQMF                  | CL10      | Identifies the current release of QMF. This value is QMFV12R1.0, and does not change during a session.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| XCBQRYL2                | F         | Contains the length of the SQL query addressed by field XCBQRYP2.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

Table 69. Fields of the DXEXCBA interface control block to the governor (continued)

| Field    | Data type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XCBQRYP  | A         | <p>Contains the address of a copy of the query that QMF passes to the database for execution. The query length is the first halfword.</p> <p>If you are writing a governor exit routine in QMF Version 12.1 for the first time, use fields XCBQRYP2 and XCBQRYL2 instead of this field to point to the SQL query and store its length, because those fields support queries of any size.</p> <p>If the query referenced by XCBQRYP is greater than 32 KB, the query is truncated at 32 KB and field XCBQRYPT is set to indicate the longer query. In this case, field XCBQRYP2 addresses the query and field XCBQRYL2 provides its length. The DSQEC_SQLQRYSZ_2M global variable controls whether SQL queries longer than 32 KB are supported.</p> <p>The governor inspects the query upon a call to start database activity (before any data retrieval) and determines whether to cancel the activity. The address is set to zero either at the beginning of the session or when the DATA object is reset or imported to temporary storage.</p> <p>This field contains information only when data retrieval is requested through one of the following commands; no information is provided for queries against Db2 for z/OS system tables or QMF control tables.</p> <p><b>DISPLAY TABLE</b><br/>EDIT TABLE</p> <p><b>ERASE TABLE</b><br/>EXPORT TABLE</p> <p><b>IMPORT TABLE</b><br/>PRINT TABLE</p> <p><b>RUN QUERY</b><br/>SAVE DATA</p> |
| XCBQRYP2 | A         | <p>Contains the address of a copy of the query that QMF passes to the database for execution. This field accommodates all queries regardless of length and should be used instead of XCBQRYP if you are writing a new governor exit routine. Field XCBQRYL2 provides the length of the query addressed by the XCBQRYP2 field.</p> <p>The governor inspects the query upon a call to start database activity (before any data retrieval) and determines whether to cancel the activity. The address is set to zero either at the beginning of the session or when the DATA object is reset or imported to temporary storage.</p> <p>This field contains information only when data retrieval is requested through one of the following commands; no information is provided for queries against Db2 for z/OS system tables or QMF control tables.</p> <p><b>DISPLAY TABLE</b><br/>EDIT TABLE</p> <p><b>ERASE TABLE</b><br/>EXPORT TABLE</p> <p><b>IMPORT TABLE</b><br/>PRINT TABLE</p> <p><b>RUN QUERY</b><br/>SAVE DATA</p>                                                                                                                                                                                                                                                                                                                                                                                                                  |



Table 69. Fields of the DXEXCBA interface control block to the governor (continued)

| Field    | Data type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XCBQRYPT | CL1       | <p>Indicates whether the query is greater than 32 KB, in which case the query addressed by field XCBQRYPT is truncated. Possible values are:</p> <p><b>0</b><br/>The query addressed by field XCBQRYPT has not been truncated (it is 32 KB or less).</p> <p><b>1</b><br/>The query addressed by field XCBQRYPT has been truncated (it is greater than 32 KB).</p> <p>When XCBQRYPT is set to 1, XCBQRYL2 stores the length of the query and XCBQRYPT2 contains the address of a copy of the query.</p> <p>The DSQEC_SQLQRYSZ_2M global variable controls whether SQL queries longer than 32 KB are supported..</p> |
| XCBREFR  | CL1       | <p>Indicates whether QMF refreshes the screen after returning from the governor; 1 indicates a refresh; 0 indicates no refresh.</p> <p>If your governor displays any screen information, set this field to 1.</p>                                                                                                                                                                                                                                                                                                                                                                                                  |
| XCBRELN  | CL2       | <p>Identifies the QMF release level. For QMF Version 12 Release 1, this is 19. This value does not change during a session.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| XCBRGRP  | CL16      | <p>Contains the name of the user's resource group. This value does not change during a session.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| XCBROWSF | F         | <p>Reflects the number of rows retrieved into the DATA object. Initially zero, this field changes value whenever more rows are retrieved. All data retrieval is counted whether data is retrieved from the database, sequential files, CICS temporary storage, or CICS transient data queues.</p> <p>QMF does not reset this field, but the governor can. For example, if your governor exit routine monitors the number of database rows retrieved, you can set this field to zero on the function call for the end of the command that began the data retrieval.</p>                                             |
| XCBSYST  | CL1       | <p>Identifies the current operating system. The value does not change during a session, and is usually set to 3, indicating TSO, ISPF, or native z/OS batch. Possible values are:</p> <p><b>3</b><br/>TSO, ISPF, or native z/OS batch</p> <p><b>5</b><br/>CICS</p>                                                                                                                                                                                                                                                                                                                                                 |
| XCBTRACE | CL1       | <p>Contains a value for the level of detail at which user exit activity is traced. Possible values are 0 (least detail), 1, or 2 (most detail).</p> <p>At the start of a session, the value of the TRACE field from the user's QMF profile is used here. After that, the value changes only when the user changes the value of the TRACE option.</p>                                                                                                                                                                                                                                                               |
| XCBUSER  | CL8       | <p>Contains the user's TSO logon ID (for TSO) or the user parameter on the job statement (for native z/OS batch). This field is not used in CICS; it contains blanks.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                          |

Table 69. Fields of the DXEXCBA interface control block to the governor (continued)

| Field    | Data type | Description                                                                                                                                                                 |
|----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XCBUSERS | CL2048    | Scratchpad area in which you can store results that you want the governor to save from one call to the next. It is initially set to blanks. QMF does not change this value. |

The structure of the DXEXCBA control block is shown here:

```

*****
*
*      CONTROL BLOCK NAME: DXEXCBA
*
*      FUNCTION:
*
*          THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND
*          EXIT ROUTINES.
*
*      STATUS: Version 12 Release 1 Level 0
*
*      INNER CONTROL BLOCKS: NONE
*
*****
*
DXEXCBA  DSECT
          DS      0D
XCBNAME  DS      CL8          -- CONTROL BLOCK IDENTIFICATION
          SPACE
XCBEXCTL DS      XL190       -- EXIT CONTROL
          ORG     XCBEXCTL
XCBAUTH  DS      CL8          ----- AUTHORIZATION ID
XCBUSER  DS      CL8          ----- USER ID
XCBPLAN  DS      CL8          ----- PLAN ID
          SPACE
XCBQMF   DS      CL10       ----- CURRENT VERSION/RELEASE
          SPACE
XCBRELN  DS      CL2          ----- QMF RELEASE LEVEL
          SPACE
XCBTRACE DS      CL1          ----- QMF EXIT TRACE LEVEL
XCBTOFF  EQU     C'0'        ----- NO TRACING
XCBTPART EQU     C'1'        ----- PARTIAL TRACING
XCBTFULL EQU     C'2'        ----- FULL TRACING
          SPACE
XCBSYST  DS      CL1          ----- OPERATING SYSTEM
XCBSYSTX EQU     C'3'        ----- MVS/ESA or XA (TSO,APPC, native)
XCBSYSTY EQU     C'5'        ----- CICS (MVS)
          SPACE
XCBPAD10 DS      CL4          ----- RESERVED FIELD
          SPACE
XCBNLANG DS      CL1          ----- CURRENT NATIONAL LANGUAGE
          SPACE

XCBKPARAM DS      CL1          ----- SETTING OF K PARAMETER
XCBKPARAM EQU     C'0'        ----- LATIN
XCBKPARAM EQU     C'1'        ----- DBCS
          SPACE
XCBDBMG  DS      CL1          ----- DATA BASE MANAGER
XCBDBMGS EQU     C'1'        ----- DB2 FOR SQL/DS
XCBDBMGD EQU     C'2'        ----- DB2 FOR OS/390
XCBDBMGW EQU     C'3'        ----- WORKSTATION DB2
          SPACE
XCBEMODE DS      CL1          ----- CURRENT EXECUTION MODE
XCBIACTV EQU     C'1'        ----- INTERACTIVE MODE
XCBBATCH EQU     C'2'        ----- BATCH MODE
          SPACE
XCBIACT  DS      CL1          ----- CURRENT INTERACT MODE
XCBIAICY EQU     C'1'        ----- INTERACTIVE EXECUTION
XCBIAICN EQU     C'0'        ----- NOT INTERACTIVE EXECUTION
          SPACE
XCBINCI  DS      CL1          ----- CURRENT COMMAND INTERFACE STATE
XCBINCIY EQU     C'1'        ----- COMMAND INTERFACE ACTIVE
XCBINCIN EQU     C'0'        ----- COMMAND INTERFACE NOT ACTIVE
          SPACE
XCBINPRC DS      CL1          ----- PROCEDURE OR LIST CMD EXEC STATE
XCBPRCY  EQU     C'1'        ----- RUNNING A PROCEDURE OR LIST CMD
XCBPRCN  EQU     C'0'        ----- NOT RUNNING PROCEDURE OR LIST CMD
          SPACE

```

```

XCBCVERB DS CL18 ----- CURRENT COMMAND VERB
          SPACE
XCBCAN DS CL1 ----- CANCEL CURRENT COMMAND INDICATOR
XCBCANN EQU C'0' ----- NO CANCELLATION
XCBCANY EQU C'1' ----- CANCELLATION IN PROGRESS
          SPACE
XCBACTIV DS CL1 ----- TYPE OF DATA BASE ACTIVITY
XCBOPEM EQU C'1' ----- OPEN
XCBFETCH EQU C'2' ----- FETCH
XCBPREP EQU C'3' ----- PREPARE
XCBDDESCR EQU C'4' ----- DESCRIBE
XCBCLOSE EQU C'5' ----- CLOSE
XCBEEXEC EQU C'6' ----- EXECUTE
XCBEEXECI EQU C'7' ----- EXECUTE IMMEDIATE
XCBCPAD20 DS CL9 ----- RESERVED FIELD
          SPACE
XCBRGRP DS CL16 ----- RESOURCE GROUP NAME
XCBCPAD30 DS CL22 ----- RESERVED FIELD
          SPACE
XCBCMDP DS A ----- POINTER TO ORIGINAL COMMAND STRING
* ----- WILL NOT CONTAIN PROMPT VALUES
          SPACE
XCBCMDL DS F ----- ORIGINAL COMMAND STRING LENGTH
          SPACE
XCBCQCE DS F ----- QUERY COST ESTIMATE VALUE
          SPACE
XCBBROWSF DS F ----- DATA BASE ROWS FETCHED FROM SOURCE
* ----- SET BY QMF; EXIT MAY RESET
          SPACE
XCBCQERR DS CL1 ----- QMF ERROR INDICATOR
XCBCQERRN EQU C'0' ----- NO QMF ERROR DETECTED
XCBCQERRY EQU C'1' ----- QMF ERROR DETECTED
XCBCLOC DS CL18 ----- CURRENT LOCATION NAME
XCBCQRYPT DS CL1 ----- XCBQRYPT TRUNCATION SWITCH
XCBCQRYTN EQU C'0' ----- XCBQRYPT IS NOT TRUNCATED
XCBCQRYTY EQU C'1' ----- XCBQRYPT IS TRUNCATED
XCBCPAD32 DS CL32 ----- RESERVED FIELD
XCBCQRYP2 DS A ----- POINTER TO SQL QUERY
XCBCQRYL2 DS F ----- LENGTH OF QUERY IN XCBQRYP2
          SPACE
XCBCQRYP DS A ----- POINTER TO SQL QUERY
* ----- QUERY LENGTH IS FIRST HALFWORD
* ----- QUERY TRUNCATED IF GREATER 32K
          SPACE
XCBCUCTL DS XL432 -- USER CONTROL AREA
          ORG XCBUCTL
XCBERRET DS F ----- EXIT ERROR RETURN CODE
XCBCMGTXT DS CL78 ----- EXIT ERROR MESSAGE TEXT
XCBCMSGNO DS CL8 ----- ISPF MESSAGE NUMBER
XCBCPANEL DS CL8 ----- ISPF MESSAGE HELP PANEL
XCBCLOGM DS CL1 ----- LOG MESSAGE INDICATOR
XCBCLOGMN EQU C'0' ----- QMF SHOULD NOT LOG MESSAGE
XCBCLOGMY EQU C'1' ----- QMF SHOULD LOG MESSAGE
XCBCREFR DS CL1 ----- REFRESH SCREEN INDICATOR
XCBCREFRN EQU C'0' ----- QMF DOES NOT HAVE TO REFRESH SCR
XCBCREFRY EQU C'1' ----- QMF SHOULD REFRESH SCREEN
XCBCPAD50 DS CL28 ----- RESERVED FIELD
          SPACE
XCBCUSERS DS CL2048 -- USER SCRATCH PAD AREA
XCBCPAD60 DS CL48 ----- RESERVED FIELD
XCBCAUTHL DS H ----- LENGTH OF AUTHORIZATION ID
XCBCAUTHX DS CL128 -- AUTHORIZATION ID EXTENDED
XCBCPAD70 DS CL50 ----- RESERVED FIELD

```

Figure 106. The DXEXCBA control block

## Related concepts

### Structure of the DXEGOVA control block

The DXEGOVA control block passes to the governor exit routine information about a user's resource constraints. This information is located in a resource control view called Q.RESOURCE\_VIEW.

### Addressing the resource control table

The GOVGROUP field of the DXEGOVA control block holds the value of the RESOURCE\_GROUP column of Q.RESOURCE\_VIEW, the view defined on the resource control table.

### Forward compatibility of earlier releases with QMF Version 12 Release 1

Most objects that were created under earlier releases can be used in QMF Version 12.1 without modification.

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

#### The trace facility

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

## Storing resource control information for the duration of a QMF session

You can use the information passed to the governor on the first call of a session for subsequent calls to the governor routine.

You can use the 2,048-byte scratchpad areas provided in the DXEGOVA and DXEXCBA control blocks to obtain the necessary storage to hold the resource control information. These fields can contain any information that you need to store. The information persists from one call to the governor to the next (if a CONNECT call doesn't change it).

The default governor uses the code shown here to address GOVUSERS, the scratchpad area in the DXEGOVA control block. You can use similar code to address the XCBUSERS scratchpad area in the DXEXCBA control block. WORK is the name of a DSECT, and WORKPTR is equated to general register 4. The WORK DSECT contains the definition for the fields that hold the information in the scratchpad areas.

The governor might also issue GETMAIN macros to obtain needed storage.

---

```
LA    WORKPTR,GOVUSERS
      USING WORK,WORKPTR
```

*Figure 107. Establishing addressability to the governor scratchpad area*

---

## Providing messages for canceled activities

Establish addressability to the QMF message service to provide messages for activities that have been canceled.

You can use the QMF message service to display a message to users after their commands are canceled by using the following fields of the DXEXCBA control block:

### **XCBMGTX**

Contains the message text

### **XCBERRET**

Contains the error return code

### **XCBMSGNO**

Contains the message ID for an ISPF message definition if QMF was invoked under ISPF in TSO

### **XCBPANEL**

Contains the panel ID for an ISPF message help panel if QMF was invoked under ISPF in TSO

Upon entry to the governor, XCBMGTX contains blanks, and XCBERRET contains binary zeros. The value of XCBERRET determines what message is displayed on the screen:

- If you want to use the default message, which is OK, command canceled, leave the zero value in XCBERRET.
- If you want to use the message A governor exit cancel occurred with return code `xxxxx`, use a nonzero value for XCBERRET; this nonzero value appears in the message in place of `xxxxx`.

If QMF initialization is canceled by the governor exit, the messages for XCBMGTXT and XCBERRET appear in the user's trace data rather than on the screen.

Set XCBLOGM to 1 to log a message in the user's trace data for any function call in your own governor exit routine. If the value of XCBERRET is nonzero, the default governor logs cancellation messages in the user's trace data by setting the XCBLOGM field of the DXEXCBA control block to a value of 1.

An ISPF message definition can contain longer message text and can designate a panel ID. To use the long text for a message and the designated panel for help, set XCBMSGNO with the message ID of the message definition and leave XCBMGTXT and XCBPANEL blank. If no help panel was designated in the message definition, the user receives no message help.

To override the long-message specification in a message definition, place the new message text in XCBMGTXT. To override the panel specification, place the new panel ID in XCBPANEL. Placing a panel ID in XCBPANEL also provides message help when the message definition doesn't specify a panel.

Leave XCBMSGNO blank if there is no relevant ISPF message definition. Then place the message text in XCBMGTXT, and the help panel ID, if any, in XCBPANEL. Leaving XCBPANEL blank, in this case, leaves the user without message help.

The governor can also log messages in the ISPF log file if QMF was invoked under ISPF. It can do this through the ISPF LOG service.

The trace facility writes messages to the DSQDEBUG data set at a level of detail determined by the value of the XCBTRACE field of the DXEXCBA control block. Use a value of zero for XCBTRACE if you do not want messages to be logged (although initialization errors are logged unless you do not allocate a trace data set). Use a value of 1 or 2 for the U trace option to get trace output.

The default governor does not log messages for termination function calls. Messages do not appear on screen if the command is run in batch from a QMF application.

### **Related tasks**

[The trace facility](#)

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

### **Related information**

Search for information about the ISPF LOG service the appropriate ISPF information.

## **Translating, assembling, and link-editing your governor exit routine**

Whether you're modifying the default governor exit routine or writing a routine of your own, you need to translate, assemble, and link-edit the routine.

### **Assembling and link-editing your governor exit routine for TSO, ISPF, and native z/OS batch**

You must assemble and link-edit your governor exit routine if you write your own routine or modify a copy of the default routine.

### **Procedure**

To assemble and link-edit your governor exit routine for TSO, ISPF, and native z/OS batch, follow these steps:

1. Assemble your governor exit routine.

QMF supports only Assembler language for governor code. The default governor is coded in Assembler, and the code was written for High-Level Assembler. You can review this code by printing certain members of the QMF1210.SDSQUSRE library.

2. Link-edit your governor exit routine.

- a) Place the load module for the governor in a library available to all your QMF users.

For example, you can use QMF1210.SDSQLOAD, which contains the load modules for QMF. This library can be part of the concatenation for STEPLIB.

- b) Name the module DSQUnGV1, where *n* is a 1-character identifier for the national language you are using.

This name is the name of the default module. Placing your own governor module in the QMF1210.SDSQLOAD library replaces the default module.

To avoid replacing the default module, you can rename it or move it to another library. Alternatively, you can place the module for your own governor in a different library in STEPLIB. If you place your module in a different library, be sure that your module's new library comes before QMF1210.SDSQLOAD in the concatenation sequence. If it does not come before QMF1210.SDSQLOAD, QMF calls the default module instead of your own.

- c) Be sure that the entry point for the new module is DSQUnGV1.

If your source code begins with a CSECT statement with the DSQUnGV1 label, there is nothing extra to do. If your source code does not begin with the DSQUnGV1 label, specify the entry name on the END statement in the Assembler code, or place it in an ENTRY statement in the linkage editor input. The default governor (DSQUEGV1) must run with AMODE(24) and RMODE(24), as shown here:

```
ENTRY DSQUEGV1
MODE AMODE(24),RMODE(24)
NAME DSQUEGV1(R)
```

## Results

Your own routine can run in either 31-bit (shown here) or 24-bit addressing mode. If your routine requires z/OS services that need 24-bit addressing mode (such as TPUT), then QMF handles the transfer from QMF running in 31-bit mode to the governor routine that is running in 24-bit mode and back to QMF in 31-bit mode.

```
ENTRY DSQUEGV1
MODE AMODE(31),RMODE(24)
NAME DSQUEGV1(R)
```

## Related tasks

[Translating, assembling, and link-editing your governor exit routine in CICS](#)

You must assemble and link-edit your governor exit routine in CICS if you write your own routine or modify a copy of the routine that is supplied by IBM.

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### **Translating, assembling, and link-editing your governor exit routine in CICS**

You must assemble and link-edit your governor exit routine in CICS if you write your own routine or modify a copy of the routine that is supplied by IBM.

## Procedure

To assemble and link-edit your governor exit routine for CICS, follow these steps:

1. Translate your program by using the CICS translator for Assembler.

When you translate your program, CICS supplies the standard CICS prolog (DFHEIENT), which establishes addressability and saves registers in the standard CICS working storage area. The standard prolog also provides a standard CICS epilog (DFHEIRET).

2. Assemble your program

QMF supports only Assembler-language for governor exit routines. The default governor is coded in this language; the code was written for High-Level Assembler. You can review this code by printing certain members of the QMF1210.SDSQSRE library.

3. Link-edit your governor program.

a) Place the load module for the governor in a library available to all your QMF users.

For example, you can use QMF1210.SDSQLOAD, which contains the load modules for QMF. This library must be concatenated with DFHRPL in CICS.

- b) Name the module DSQUnGV3, where *n* is a 1-character identifier for the national language you are using.

This name is the name of the default module. Placing your own governor module in the QMF1210.SDSQLOAD library replaces the default module.

To avoid replacing the default module, you can rename it or move it to another library. Alternatively, you can place the module for your own governor in a different library in DFHRPL. If you place your module in a different library, be sure that your module's new library comes before QMF1210.SDSQLOAD in the concatenation sequence. If it does not come before QMF1210.SDSQLOAD, QMF calls the default module instead of your own.

- c) Be sure that the entry point for the governor module is DSQUnGV3.

If your source code begins with a CSECT statement with this label, there is nothing else to do. If not, specify the entry name on the END statement in the Assembler code, or place it in an ENTRY statement in the linkage editor input.

- d) You must include the CICS command interface control modules DFHEAI and DFHEAI0. You must also place the control modules at the beginning of the governor load module.

In CICS, the governor must run with AMODE(31) and RMODE(31), as shown here:

```
INCLUDE SYSLIB(DFHEAI)
INCLUDE SYSLIB(DFHEAI0)
ORDER DFHEAI,DFHEAI0
ENTRY DSQUEGV3
MODE AMODE(31),RMODE(31)
NAME DSQUEGV3(R)
```

### Related tasks

[Assembling and link-editing your governor exit routine for TSO, ISPF, and native z/OS batch](#)

You must assemble and link-edit your governor exit routine if you write your own routine or modify a copy of the default routine.

## Canceling user activity

You have multiple options for canceling user activity, including the QMF cancellation service and the QMF interrupt facility.

### QMF cancellation service

QMF provides a cancellation service that can be used to cancel a QMF command or database activity. When the QMF cancellation service is called, QMF turns on an internal cancel switch that is checked by the QMF command or database activity. If QMF is able to cancel the activity, a cancel message is displayed. If the QMF activity is about to complete, the cancellation request might be ignored and the activity might be allowed to complete.

To have your governor call the QMF cancellation service to cancel an activity, branch to the address that appears in the GOVCADDR field of the DXEGOVA control block. [Figure 108 on page 313](#) shows the statements that establish addressability to the QMF cancellation service. Ensure that register 13 points to a save area for the governor so that QMF cancellation service can restore the state of the governor when it returns control.

```
L R15,GOVCADDR
   BALR R14,R15
```

*Figure 108. Calling the QMF cancellation service*

The cancellation routine returns control to the point that is addressed by register 14 (in this case, the command that follows the BALR command). Register 15 contains a return code. [Figure 109 on page 314](#) contains possible return codes.

```

0 - QMF accepted the cancel request.
100 - QMF is not active.

```

*Figure 109. Cancellation service return codes*

The following governor functions can be canceled using the QMF cancel service:

| Governor Function Code * | Activity Being Performed   |
|--------------------------|----------------------------|
| GOVSCMD (3)              | Start of a QMF command     |
| GOVSDBAS (6)             | Start of database activity |

\* Governor function codes are located in the DXEGOVA control block.

### Asynchronous timer routine

The cancel service can be called from a timer routine if the timer routine can access the address of the QMF cancel service provided in the GOVCADDR field of the DXEGOVA control block.

**Important:** The timer routine cannot be used to cancel an idle QMF session. When QMF is waiting for user input, the timer routine cannot complete until the user provides input to the screen or the QMF attention handler is activated by the ATTN key.

### z/OS services

The QMF governor exit routine can use z/OS services, such as ABEND to terminate a QMF session.

### Cancelling an idle QMF session

To cancel a QMF session that is idle and waiting for input, you can use the Job Wait Time (JWT) option of the z/OS system management facilities parameters (SMFPRMxx) member of the SYS1.PARMLIB. For more information, see the information on the Time Limit Exit in the z/OS information.

### Freeing locks held by an idle QMF session

To cancel a database thread that is holding locks while the QMF session is idle, see the information on timing out idle active threads in the Db2 for z/OS information.

### QMF interrupt facility

The QMF interrupt facility can be used to attempt to cancel a command that is taking too long to run. For more information, see [“Creating an interrupt to capture diagnostic information” on page 354](#).

## Using the Db2 resource limit facility

Db2 has its own governor, or resource limit facility. You can use the Db2 resource limit facility with either the QMF for TSO or the QMF High Performance Option governing functions. You can also use the Db2 governor independently.

You can control all access to the database, as well as distributed access, with the Db2 resource limit facility. And you can use the facility to monitor the processor time used for dynamically run SELECT, INSERT, UPDATE, and DELETE statements.



## Differences between governors

You can supplement the operations of the QMF governor with the Db2 governor. However, the Db2 governor differs from the QMF governor in what and how it monitors.

The QMF and Db2 governors differ in these ways:

- The Db2 governor limits its monitoring to dynamic SELECT, INSERT, UPDATE, and DELETE statements. It does not monitor, for example, the processor time spent in running a CREATE or DROP statement.
- The Db2 governor limits its monitoring to processor time. It does not count row fetches, as the QMF governor does.
- Processor time for the Db2 governor includes only the time consumed by Db2. In contrast, the QMF governor includes the time QMF spends running a command—manipulating a spill file, for example, or displaying the first page of the results of running a SELECT query.
- When a user runs a SELECT query, the Db2 governor monitors all the time Db2 spends running the query, beginning with the PREPARE statement and continuing through the row fetches and the closing of the cursor. The QMF governor ends its monitoring after the first page of the results are displayed. Any subsequent row fetch is treated as part of the scrolling command that caused the fetch to occur.
- The Db2 governor makes no provision for a cancellation prompt; its only control parameter for a given QMF session is maximum processor time.

## QMF commands that can be monitored by the Db2 governor

You can monitor QMF commands with the Db2 governor.

Each row in the following table represents a group of QMF commands that can be individually monitored by the Db2 governor. Read the description of each package in the third column of the table to determine the proper package to govern according to your QMF configuration.

Package names for which the descriptions refer to systems other than Db2 for z/OS are used to govern SQL statements at remote servers, which are accessed by the QMF CONNECT command or QMF commands that include three-part names. To use the Db2 resource limit facility to govern SQL statements at remote servers, additional configuration steps are required to set up governing.

| QMF command                                                                                                                                                                                                                                                                                              | Type of statement governed | QMF package names to govern                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• RUN QUERY (QBE P. queries, prompted queries, and SQL SELECT queries)</li> <li>• DISPLAY TABLE (CONFIRM=YES)</li> <li>• EXPORT TABLE</li> <li>• PRINT TABLE</li> <li>• Commands that scroll reports (BOTTOM, TOP, FORWARD, BACKWARD, RIGHT, and LEFT)</li> </ul> | SELECT                     | <ul style="list-style-type: none"> <li>• DSQJFSUS - controls single row fetch operations, which apply when QMF is started in a Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows database with the MR program parameter set to NO</li> <li>• DSQJFSUM - Controls multirow fetch operations, which apply when QMF is started in a Db2 for z/OS database with the MR program parameter set to YES</li> <li>• DSQJFSUV - Controls fetching of rows from a DB2 for VSE and VM database</li> </ul> |

Table 70. Groups of QMF commands that can be individually monitored by the Db2 governor (continued)

| QMF command                                                                                            | Type of statement governed | QMF package names to govern                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• SAVE DATA</li> <li>• IMPORT TABLE</li> </ul>                  | INSERT                     | <ul style="list-style-type: none"> <li>• DSQJSDT7 – Controls single row insert operations, which apply when QMF is started in a Db2 for z/OS, DB2 for iSeries, or Db2 for Linux, UNIX, and Windows database with the MR program parameter set to NO.<br/><br/>Single row insert is also used when a SAVE DATA command is issued to save a table that contains XML or LOB data.</li> <li>• DSQJSDT8 – Controls multirow insert operations, which apply when QMF is started in a Db2 for z/OS database with the MR program parameter set to YES</li> </ul> |
| RUN QUERY (QBE I., U., or D. queries or SQL queries that include INSERT, UPDATE, or DELETE statements) | INSERT<br>UPDATE<br>DELETE | <ul style="list-style-type: none"> <li>• DSQJESQL – For DB2 for VSE and VM databases</li> <li>• DSQJESQM – For all other database types</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                       |
| EDIT TABLE (MODE=ADD)                                                                                  | INSERT                     | DSQJTSQL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| EDIT TABLE (MODE=CHANGE, SAVE=IMMEDIATE)                                                               | UPDATE<br>DELETE           | DSQJHSQL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| EDIT TABLE (MODE=CHANGE, SAVE=END)                                                                     | UPDATE<br>DELETE           | DSQJNSQL (DB2 for VSE and VM only)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

### Related concepts

#### Configuring Db2 governing of QMF commands

You configure the governing of QMF commands directed to the local Db2 for z/OS database by adding rows to a specific resource limit specification table (RLST).

#### Enabling support for multirow fetch and insert

The DSQSMRFI parameter controls whether the database uses multirow or single-row fetch and insert.

## How QMF responds when queries are cancelled by the Db2 governor

When a query exceeds the maximum processor time, the Db2 governor ends the query and returns an SQL error code of -905, signaling to QMF that the query has been canceled.

How QMF handles the cancellation depends on where in a QMF session the Db2 governor canceled the query:

### **During QMF initialization**

When it begins a user's session, QMF runs several queries that the Db2 governor monitors. If any of these queries are canceled, QMF ends the session. Before the session ends, QMF writes an explanatory record in the trace data.

For example, suppose that you want to restrict QMF users to using QMF during certain periods of time. During these periods of time, you can assign restricted users a maximum processor time of zero so that the QMF session is cancelled before the QMF home panel is displayed.

### **After QMF initialization**

After initialization, QMF treats the cancellation of a query just as it treats any other error in running a query. Suppose, for example, that the Db2 governor cancels an INSERT statement. The database

changes resulting from the INSERT statement are rolled back and QMF issues an error message. If the user then asks for message help, a panel explaining the governor's action is displayed.

Suppose, instead, that a cancellation takes effect while the user is scrolling through a report. In this case, it is likely that a row fetch caused the cancellation. The cancellation leaves the DATA object incomplete. Because Db2 closes the cursor, the DATA object cannot be completed and a RESET DATA command must be issued.

## Configuring Db2 governing of QMF commands

You configure the governing of QMF commands directed to the local Db2 for z/OS database by adding rows to a specific resource limit specification table (RLST).

When the Db2 for z/OS subsystem in which QMF is installed is started, the subsystem is associated with a specific RLST. An RLST provides the Db2 governor with input for all the users in the subsystem with which the RLST is associated, including QMF users.

Different RLSTs can be associated at different times with the same Db2 subsystem. For example, one RLST for a particular period of time might make it impossible for users to start a QMF session during that period of time; in these cases, any attempt to start a QMF session ends during QMF initialization, and a message appears in the trace data. You can use different RLSTs to handle site-specific rules and restrictions or provide governing for individual users or groups of users. For example, you can add rows for a few individual users and a row that applies to everyone else. The rows for the individual users contain their primary authorization IDs. The row for the other users contains a blank value for the authorization ID.

To configure governing of QMF commands directed to the local Db2 for z/OS database, add one row to the Db2 RLST for each package shown in [“QMF commands that can be monitored by the Db2 governor” on page 315](#) that you want to govern. The columns to be updated in the RLST depend on whether you want to govern QMF commands predictively or reactively. More than one row can be associated with a QMF package name if both predictive and reactive governing are required for the package.

To use the Db2 resource limit facility to govern SQL statements at remote servers, additional configuration steps are required.

### How to update RLSTs for predictive governing

For predictive governing, update the following columns of the RLST table:

| <i>Table 71. Values to add to the Db2 for z/OS RLST table to predictively govern QMF commands</i> |                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Column of RLST table</b>                                                                       | <b>Value</b>                                                                                                                                                                                     |
| RLFPKG                                                                                            | The name of the package associated with the QMF commands to be governed<br><br>Package names are shown in <a href="#">“QMF commands that can be monitored by the Db2 governor” on page 315</a> . |
| RLFFUNC                                                                                           | 7                                                                                                                                                                                                |
| RLFASUERR                                                                                         | User-determined                                                                                                                                                                                  |
| RLFASUWARN                                                                                        | User-determined                                                                                                                                                                                  |
| RLF_CATEGORY_B                                                                                    | User-determined                                                                                                                                                                                  |

### How to update RLSTs for reactive governing

For reactive governing, update the following columns of the RLST table:

Table 72. Values to add to the Db2 for z/OS RLST table to reactively govern QMF commands

| RLST column | Value                                                                                                                                                                                        |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RLFPKG      | The name of the package associated with the QMF commands to be governed.<br>Package names are shown in <a href="#">“QMF commands that can be monitored by the Db2 governor”</a> on page 315. |
| RLFFUNC     | 2                                                                                                                                                                                            |
| ASUTIME     | Maximum processor time to be allocated to completing the QMF command associated with the package in RLFPKG                                                                                   |

**Important:** Do not update the RLFCOLLN column of the RLST table. Attempting to govern QMF functions by the QMF collection ID (Q) can stop QMF internal processing in certain situations or otherwise cause QMF to function improperly.

---

## Chapter 18. Running QMF in batch mode

If a user runs a procedure with the RUN command, the user cannot execute QMF commands except to cancel the procedure or the session. Thus, running a procedure using the RUN command can tie up considerable session time.

Alternatively, you and your users can create and run procedures and applications in batch mode, in which case they run independently of the user's session so that the user can continue to issue commands.

If you need to run QMF noninteractively and you use QMF for TSO, you might consider starting QMF as a Db2 for z/OS stored procedure. This interface allows any program that can call a Db2 for z/OS stored procedure, such as QMF for Workstation, to start QMF for TSO as a stored procedure and receive output back in a result set. You can also run a stored procedure to start a batch job from a remote Db2 client.

If you are using an NLF, users at a multilingual site can choose the language environment for their batch QMF sessions, just as they can for their interactive sessions.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

---

## Running QMF as a batch program on z/OS

You can use QMF batch mode in TSO, ISPF, and native z/OS. For ISPF and TSO, the batch facility executes QMF in the TSO terminal monitor program (TMP).

You need proper authority to operate in batch mode. You also need to understand security implications, debugging procedures, and how to send a batch job.

### Authority to operate in batch mode

Certain QMF and Db2 authorizations are required to submit a batch job.

To begin granting these authorizations to users who need them, first determine the logon ID and Db2 primary authorization ID under which your job will be running:

- If your site uses RACF, the logon ID is the value of the USER parameter on your JOB statement. The Db2 primary authorization ID is the one corresponding to that logon ID.
- If your site does not use RACF, the logon ID and primary authorization ID are determined in the JCL to execute a batch job.

The logon ID and authorization ID play the same role as when you use QMF interactively. As a result, the procedure runs only if the following conditions are satisfied:

- You can operate QMF interactively using the logon ID for the batch run.

Users with authority to use QMF interactively and run jobs in the background can also use it in batch mode.

- The authorization ID corresponding to the logon ID owns the procedure to be run, or that procedure is shared.

In running the procedure's commands, the authorization ID works interactively. However, not every QMF command that can be run interactively can be run in batch mode.

### RACF security considerations

If RACF is a part of your security, you can prevent users from running jobs under other users' logon IDs. A user who runs such a job can access all the Db2 data that the other user has access to, including data that the user running the job is not authorized to see.

## JCL to execute a QMF batch job

You or your users must create a JCL file that runs the statements.

Batch-job JCL is similar to TSO logon JCL because QMF is run in batch mode through batch-mode TSO.

- The job statement

Begin your JCL with a JOB statement like this:

```
//BATCH JOB USER=LMN,PASSWORD=ABC,NOTIFY=LMN
```

The statement that is shown might not be adequate for every site because it does not contain accounting information or the user's name. The operands that are shown specify that:

- The logon ID is LMN.
- The logon password is ABC.
- The terminal message is sent to user LMN when the job ends.

You can include other operands, including the MSGLEVEL and MSGCLASS operands that control the level of detail and the routing of the JCL and system messages.

**Important:** Without RACF, the PASSWORD parameter is ignored, creating a security exposure.

- The EXEC statement

You can use an EXEC statement for a JOB step to run batch-mode QMF similar to the following:

```
//SAMPLE EXEC PGM=IKJEFT01,TIME=1440,DYNAMNBR=30,REGION=3072K
```

This statement:

- Calls TSO (PGM=IKJEFT01)
- Specifies an adequate number of allowable dynamic allocations (DYNAMNBR=30)
- Specifies a sufficiently large region for QMF (REGION=3072K)

- The DD statements

You can use the same DD statements both for running QMF interactively and for batch mode. You must remove the statements for SYSPRINT, SYSTEMR, and SYSIN.

You can add the operand HOLD=YES to one or more of the SYSOUT DD statements and then manipulate their output with the TSO OUTPUT command (another FIB command). Using the OUTPUT command, you can route the output of the SYSOUT DD statement to your screen.

You also need DD statements for the SYSTSPRT and SYSTSIN data sets, as follows:

### SYSTSPRT

This data set contains the message output from TSO and ISPF. For this data set, specify DD statements like those shown here:

```
//SYSTSPRT DD SYSOUT=A
```

### SYSTSIN

SYSTSIN holds the TSO statements that run during the job step. To include these statements in your JCL, specify DD statements like those shown here:

```
//SYSTSIN DD *
EXEC CLISTA
PROFILE PREFIX(LMN)
ISPSTART PGM(DSQQMF) NEWAPPL(DSQE) PARM(DSQSMODE=B,DSQSRUN=LMN.PROCA)
:
/*
```

TSO runs these statements in their order of appearance in SYSTSIN:

- The first statement runs a CLIST named CLISTA, which can allocate QMF libraries.
- The second statement sets the user's dsname prefix to LMN.

- The ISPSTART statement invokes batch-mode QMF with ISPF and runs the procedure LMN.PROCA.
- The PROFILE PREFIX statement
 

The PROFILE PREFIX statement sets the user's dsname prefix to LMN, which is assumed in the example to be the user's logon ID.

Place the PROFILE PREFIX statement before the first ISPSTART statement that starts QMF. Issuing the PROFILE PREFIX statement within QMF is ineffective.

The PROFILE PREFIX statement can make permanent changes to the user's TSO profile, depending on your site's setup. If it does, a user might want to restore the dsname prefix. The initial value of the prefix setting is in the ISPF system variable ZPREFIX.

For the PROFILE PREFIX statement to be effective, the DSQSPRID parameter must be set to TSOID. A similar statement (one setting the user's prefix to the user's logon ID) might be needed in other jobs that run QMF in batch mode for the following reasons:

- To identify the user to QMF when RACF is not used

At sites where RACF is not used, QMF assumes that the user's logon ID is equal to the user's dsname prefix. If this prefix is null, QMF assumes that the logon ID is BATCH. Thus, by setting the dsname prefix to the user's logon ID, the PROFILE PREFIX statement provides the user's logon ID to QMF.

The primary authorization ID that Db2 assigns to the user in this case is the value that is specified by the Db2 installation parameter UNKNOWN AUTHID. The logon ID is used in trace output that is recorded in the DSQDEBUG data set. Either the primary authorization ID or the logon ID is used for reading from the profile and assigning a default resource group, depending on the setting of the DSQSPRID parameter.

- To avoid problems with data set names

You can encounter problems when a QMF procedure uses both the fully qualified and the incomplete forms of a data set's name in the QMF IMPORT/EXPORT commands. For example, consider a procedure that runs under the logon ID LMN, which issues the following two commands:

```
EXPORT QUERY TO 'LMN.QUERYX.QUERY'
IMPORT QUERY FROM QUERYX
```

The EXPORT command uses the logon ID (LMN) as the first qualifier for the name of the data set containing the exported objects. The IMPORT command then imports the contents of this data set.

If the user's dsname prefix is ABC instead of LMN, the file that is referenced in the IMPORT statement is named 'ABC.QUERYX.QUERY' instead of 'LMN.QUERYX.QUERY'. This is because the prefix is used for the first qualifier of a data set name when, as in the example IMPORT command, the name is not fully qualified.

In these cases, the procedure cannot find the file that it previously exported. The PROFILE PREFIX statement avoids this problem by setting the dsname prefix to the user's logon ID (in this case, 'LMN').

## Related concepts

### Setting program parameters and preferences at startup time

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

### Specifying which ID to use as the QMF profile key under TSO

If you choose to create specific profiles for each user in QMF for TSO, the values in the CREATOR column of the table can all be TSO logon IDs or Db2 primary authorization IDs. If you choose to have QMF authenticate users using their TSO logon IDs, you must specify that choice when you start QMF.

## Running QMF batch in native z/OS

In addition to running QMF batch in TSO and ISPF, you can run QMF as a native z/OS batch job.

You can use JCL like the following example.

```

/*****/
//QMFBAT JOB 00299000
//S1 EXEC PGM=DSQMFE,PARM='M=B,I=yourQMFproc' 00300000
//* 00301000
//* Program libraries required when running in batch 00302000
//* 00303000
//STEPLIB DD DSN=QMF1210.SDSQLOAD,DISP=SHR 00304000
// DD DSN=DSN1110.SDSNEXIT,DISP=SHR 00305000
// DD DSN=DSN1110.SDSNLOAD,DISP=SHR 00306000
// DD DSN=GDDM.ADMLoad,DISP=SHR 00307000
//* 00308000
//* QMF/GDDM maps are required when running in batch 00309000
//* 00310000
//ADMGGMAP DD DSN=QMF1210.SDSQMAPE,DISP=SHR 00311000
//* 00312000
//* 00313000
//* 00314000
//* Data sets used by QMF 00315000
//* 00316000
//DSQPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330) 00317000
//DSQDEBUG DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210) 00318000
//DSQDUMP DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=125,BLKSIZE=1632) 00319000
//DSQSPILL DD DSN=&&SPILL,DISP=(NEW,DELETE), 00320000
// UNIT=SYSDA,SPACE=(TRK,(100),RLSE), 00321000
// DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096) 00322000
//* 00323000
/*****/ 00324000

```

Figure 110. JCL to run QMF as a native z/OS batch job

When you run QMF in native z/OS, remember that:

- TSO is not available.
- QMF functions that require TSO or ISPF do not work when you run QMF in native z/OS.
- The default user ID suffix is not available; you must use the fully qualified data set names to export or import files.
- You cannot use procedures with logic (REXX procedures). To run QMF with REXX in a non-TSO address space, you must use IRXJCL, as illustrated in [Figure 111 on page 323](#).

The REXX program that is listed here uses the QMF callable interface to start QMF and run QMF commands in batch mode.



```

//QMFBATCH JOB REGION=8M,
//  MSGCLASS=H, TIME=(2,30), USER=&SYSUID, NOTIFY=&SYSUID, CLASS=A
//ROBQMF1 EXEC PGM=IRXJCL
//STEPLIB DD DSN=DSN1110.DB2A.SDSNLOAD, DISP=SHR
// DD DSN=DSN1110.DB2A.SDSNEXIT, DISP=SHR
// DD DSN=QMFDEV.QMF1210.SDSQLOAD, DISP=SHR
//ADMGGMAP DD DSN=QMFDEV.QMF1210.SDSQMAPE, DISP=SHR
//SYSEXEC DD DSN=ROBIN.QMF1210.SDSQEXCE, DISP=SHR
//DSQPRINT DD SYSOUT=*, DCB=(RECFM=FBA, LRECL=137, BLKSIZE=1330)
//DSQDEBUG DD SYSOUT=*, DCB=(RECFM=FBA, LRECL=121, BLKSIZE=1210)
//DSQDUMP DD SYSOUT=A, DCB=(RECFM=VBA, LRECL=125, BLKSIZE=1632)
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//DSQSPILL DD DSN=&&SPILL, DISP=(NEW,DELETE),
// UNIT=VIO, SPACE=(CYL,(10,20),RLSE),
// DCB=(RECFM=F, LRECL=4096, BLKSIZE=4096)
//SYSTSIN DD *
/* REXX */
CALL DSQCIX "START (DSQSMODE=BATCH"
SAY DSQ_MESSAGE_ID DSQ_MESSAGE_TEXT
IF DSQ_RETURN_CODE = DSQ_SEVERE THEN EXIT DSQ_RETURN_CODE
CALL DSQCIX "RUN PROC REXXPP"
SAY DSQ_MESSAGE_ID DSQ_MESSAGE_TEXT
CALL DSQCIX "EXIT"
SAY DSQ_MESSAGE_ID DSQ_MESSAGE_TEXT
EXIT DSQ_RETURN_CODE
/*

```

Figure 111. REXX program to start and run QMF in batch mode

These JCL samples allocate a spill file. Instead of allocating a file for spill data, you can use the DSQSPTYP parameter to spill data to extended storage.

### Related tasks

Spilling report data to extended virtual storage (TSO only)

In QMF for TSO, use extended storage for spill data unless the system on which QMF is running has very limited extended storage available.

## Running QMF batch under TSO

You send a batch job to z/OS by using the TSO SUBMIT command.

You or your users must create the QMF procedure to be run and save it in the database. The procedure might issue queries or run other procedures and might run most other QMF commands. Through the TSO command of QMF, the procedure might also call CLISTS or online programs.

After you save the procedure, you or your users must create a JCL file for the job that runs the procedure. The JCL for this job calls TSO for batch operations. It must allocate resources that TSO and QMF need, including a data set containing statements that TSO is to run. One of these statements must start a QMF session.

Submit the job to the background through the TSO SUBMIT command. SUBMIT is one of the FIB (foreground-initiated background) commands through which the user runs, monitors, and manipulates background jobs. Issuing an FIB command requires the proper TSO authority. Granting this authority is a TSO administration task.

The SUBMIT command can be run:

- During the user's QMF session by using the QMF TSO command
- In TSO READY mode or in a CLIST that customizes the JCL of the job

The customization can be based on parameters whose values are passed to the called CLIST.

Any error that is encountered while the procedure runs can:

- Terminate the procedure
- Back out an uncommitted Db2 unit of recovery

The JOB statement for the job can specify that a message is sent to the user when the job is done. The message appears on the user's screen. The user does not need to end the QMF session to receive the message.

After the run ends, the user can examine the job output for errors. With the proper JCL, this output can be routed to data sets that the user can print or examine with an editor. One of these data sets can contain a record of the confirmation and error messages and, if wanted, a record of the QMF commands that ran.

## Running QMF batch under ISPF using the QMF BATCH command

The QMF batch query/procedure application is designed to minimize the amount of effort involved and knowledge required to run a query or procedure in batch mode. To use the application, you must start QMF under ISPF.

If you are using an NLF, you need to assign a translated synonym for the batch application to the users. They then issue the translated command synonym for BATCH.

### Related tasks

[Customizing Command synonyms](#)

If the default command synonyms do not meet your needs, you can create your own synonyms. After you create a command synonym, users can enter the synonym on the command line in the same way that they enter a QMF command.

### Authority to use the batch application in ISPF

The batch query/procedure application creates the procedure and JCL for the user's batch job, but it is not able to submit the job unless the user has authority to use the TSO FIB (foreground-initiated background) commands. A TSO administrator grants the user this authority.

The batch job is run under the user's TSO logon ID, so the commands issued by the batch procedure are run under the user's authorization ID. The same rules apply to a batch job and to the user running the job interactively:

- If the query, procedure, or form to be run is not owned by the user, it must be shared by its owner.
- For any table referenced in the query (assuming a retrieval query), the user must have the SQL SELECT privilege.
- If the query or procedure results are to be saved in a new table, you must enable users to create tables.

### Related concepts

[Enabling users to create tables in the database](#)

Depending on the needs of your site, you might need to create tables for your users or enable them to create their own tables.

### Using the batch application in ISPF

Any QMF user can use the batch query/procedure application, because starting it consists of running a shared procedure.

Before starting the batch query/procedure application, the user must have the query or procedure available to be run, and, if necessary, a form to format the report. These objects can be either in the database or in temporary storage. If the objects are in the database, they can be owned by others, provided they are shared.

After the user completes the appropriate fields and presses Enter, the application composes a batch job and submits it to the background.

While the prompt panel is displayed, the user can:

- Display the application's help panels by pressing the Help function key
- Terminate the application by pressing the End function key

(The function key settings appear at the bottom of the prompt panel.)

If you are using an NLF, issue the translated command synonym for BATCH to run a query or procedure in batch mode. For example, the translated German command synonym for BATCH is STAPEL. For the

translated command synonym for BATCH in the other language environments, see the Q.COMMAND\_SYNONYM\_n control table, where n is a 1-character language identifier.

## Related tasks

### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### **Starting the batch application in ISPF**

The advantage to using the batch application is the BATCH command prompt panel because the user does not need to know JCL or QMF procedures. The user outlines what the job should do in the panel and leaves the details of how to do it to the application.

### **About this task**

The batch query/procedure application must be invoked while its user is operating under QMF. When invoked, the application prepares a batch job for the user and submits it to the background. The job is prepared from information the user enters on a prompt panel. It runs a single query or procedure of the user's choice. Assuming the batch job is a SELECT query, the job can also:

- Save the DATA object that is created by running the query
- Format the report object using a form of the user's choice
- Print the report
- Write the report to a permanent data set
- Send the report to one or more other users

### **Procedure**

- To use the batch application, enter:

```
BATCH
```

### **Results**

This command results in the display of the QMF BATCH command prompt panel as shown here:

```
                QMF Batch Command Prompt
Enter name of Query or Procedure ==>
  Use Object from work area?    ==>  NO
  Is Object QUERY or PROC      ==>  QUERY
  PROC arguments ==>
Enter Name of FORM to use      ==>
  Use FORM from work area?     ==>  NO
Enter Name of QMF Batch PROC ==>
Enter Name of Data to Save     ==>
Enter Name of Report Dataset ==>
  Is Dataset New?              ==>
  Enter Optional Volser        ==>
Enter Report Width              ==> 133
Should Report be Printed?      ==> YES
Enter Output Class              ==>  A
Log Messages and Commands?     ==> YES
Enter Userids and Nodes to Send Report:
  Userid(1)                    ==>      Node      ==>
  Userid(2)                    ==>      Node      ==>
Enter Database SUBSYSTEM ==>      Enter Database PLANID ==>
Enter TSO Logon Password ==>      ( Press ENTER Key to Process Request )
Please enter a name for the batch ITEM.
```

Figure 112. The QMF BATCH command prompt panel

### The fields on the batch command prompt panel

The fields on the batch prompt panel describe what the batch job should do.

### Required entry fields

Certain fields on the batch prompt panel are mandatory. QMF prompts you for values for these required fields if necessary. The following table describes the required fields.

| <b>Field text</b>                | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enter name of Query or Procedure | A value is required for the name of the query or procedure to be run in batch mode. If the query or procedure is currently in temporary storage, it is saved in the database using this name. If the name is that of an existing object, the new object replaces the old one. (The name must be unqualified.) If the object is in the database, enter the name under which it was saved. (The name must be qualified if the object is owned by someone else and shared.) Save this object using CONFIRM = NO as a profile setting.                                                                                                               |
| Is Object QUERY or PROC          | The object type to be run in batch; must be either QUERY or PROC.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Enter Name of QMF Batch PROC     | A value is required for the name of the QMF procedure to be run in batch mode. (The name is not qualified.) If you are submitting multiple queries, you need to modify the Enter name of Query or Procedure field for each query or the new batch job replaces the old job. This procedure contains the appropriate QMF commands depending upon user input. The query or procedure that was specified in the Is Object QUERY or PROC field is run from this procedure. The procedure is saved using the SHARE = YES keyword option. It must be able to be run by the batch machine. Save this procedure using CONFIRM = NO as a profile setting. |

### Optional entry fields

The following table describes the remaining entry fields on the panel, which are optional. Where a value of YES or NO is expected, a default YES or NO normally appears on the screen. If you delete a value in a YES/NO field, QMF prompts you for an entry.

| <b>Field text</b>          | <b>Description</b>                                                                                                                                                                                                                                         |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Use Object from work area? | If the batch query or procedure is currently in temporary storage, enter YES in this field. The query or procedure is then saved to be run later in batch. If the query or procedure is in the database, enter NO. The default value for this field is NO. |
| PROC arguments             | Arguments to the REXX procedure named in the Enter name of Query or Procedure field.                                                                                                                                                                       |

Table 74. Optional entry fields for the QMF BATCH application (continued)

| Field text                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enter Name of FORM to use    | <p>To run the batch query using a form, you must specify the name of a form in this field. If the form to be used:</p> <ul style="list-style-type: none"> <li>• Is the default form, leave the field empty.</li> <li>• Is in the database, the form is saved using this name. The name must be qualified if the form is owned by someone else and shared.</li> <li>• Is the current form, enter a name under which it can be saved. The name must be unqualified, because the form is saved under your own authorization ID.</li> </ul> <p>This form is saved using CONFIRM = NO as a profile setting.</p> <p>If you enter the name of an existing form, the new form replaces the old.</p> |
| Use FORM from work area?     | <p>If the batch form is the current form, enter YES in this field. The form is then saved for use later in batch. If the form is in the database, enter NO. The default value for this field is NO.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Enter Database SUBSYSTEM     | <p>Enter the name of the Db2 subsystem that QMF uses; this field has the same value as program parameter DSQSSUBS.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Enter Database PLANID        | <p>Enter the name of the QMF application plan; this field has the same value as program parameter DSQSPLAN. The default is QMF12.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Enter TSO Logon Password     | <p>Enter your logon password; it does not appear on the screen.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Log Messages and Commands?   | <p>The default value for this field is YES. This means that the default trace level in batch mode is L2, which traces messages and commands. If you do not want tracing at the L2 level, specify NO. Tracing does not continue in the batch procedure beyond the SET PROFILE (TRACE=NONE command, which is then in the generated user procedure.</p>                                                                                                                                                                                                                                                                                                                                        |
| Enter Name of Data to Save   | <p>If you want the data resulting from running a query or procedure to be saved, a value must be given for this field. The DATA object is saved as a new table using this name and the CONFIRM=NO keyword option.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Enter Name of Report Dataset | <p>If you want the report to be written to a permanent data set, enter the name of that data set here. The name must be fully qualified. If you do not want this done, leave the field empty.</p> <p>This data set name is passed to z/OS via JCL statements and must conform to the z/OS naming conventions. Fully qualified names do not require quotation marks if the name does not contain any special characters other than period, @, #, or \$. If quotation marks are used, z/OS assumes that special characters are used and does not catalog the data set in the system catalog.</p>                                                                                              |
| Is Dataset New?              | <p>You must enter something in this field if you entered a data set name in REPORT DATASET. Enter YES to show that this data set does not currently exist. Enter NO to show that the data set does currently exist.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Table 74. Optional entry fields for the QMF BATCH application (continued)

| Field text                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enter Optional Volser                   | Optional: Fill in this field if you entered YES in the Is Dataset New? field. Enter the serial number of a volume on which the new data set can reside. The volume must be one that can be used on a unit of the SYSDA class, as defined by your installation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Enter Report Width                      | If you entered YES in the Is Dataset New? field, you must complete this field. Its value becomes the logical record length (LRECL) of the new data set. If the width of your report is less than or equal to the LRECL, use the default value of 133.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Should Report Be Printed?               | This field must contain YES or NO. YES indicates to print the job; NO indicates to not print the job.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Enter Output Class                      | Enter the output class for the printed output from your job. The printed output includes: <ul style="list-style-type: none"> <li>• The system messages</li> <li>• The report (DSQPRINT), if it was printed</li> <li>• The DSQDEBUG trace data set</li> <li>• An abend dump (DSQUDUMP), if one was produced</li> </ul> If your installation provides for it, you can choose an output class that holds the printed output for routing to your display device.                                                                                                                                                                                                                                                                                     |
| Enter Userids and Nodes to Send Report: | If you want the resulting report to be sent to other users, enter their user IDs and nodes in these fields. To use the fields, you need to name a data set for the report output in the Enter Name of Report Dataset field. <p>On the same line, enter a user's logon ID in one of the Userid fields and the user's node in the corresponding Node field. In this way, you can specify up to two recipients for the report. The report is sent using the TSO TRANSMIT command. You do not need to complete the Node field for a user if that information is in your NAMES.TEXTLIST data set. The node ID you specify might correspond to an entire list of names in this file, allowing you to send the report to more than just two people.</p> |

### Modifying the batch application

You can modify the batch application by making changes to its components or creating new components for the customized application. Create new components so that you do not risk losing your changes when maintenance is performed.

### Applicable QMF components

To modify the batch application, you need to be aware of the following components in the QMF libraries:

- The CLISTs DSQABB11 and DSQABB12 in the QMF1210.SDSQCLTE library

When users call the batch application with the BATCH command, they actually call DSQABB11. The purpose of this CLIST is to call DSQABB12 through the ISPF SELECT service as a new application. Most of the logic in the application is in DSQABB12.

- ISPF message definitions in members DSQBE00, DSQBE01, and DSQBE02 of the QMF1210.SDSQMLBE library

These messages appear on the user's screen after the application ends. The application generates these messages using the QMF MESSAGE command.

- Various ISPF panel definitions in the QMF1210.SDSQPLBE library, which serve a variety of purposes:
  - DXYEABMP is the application's prompt panel.
  - DXYEABM1, DXYEABM2, and DXYEABM3 are the help panels for the prompt panel.
  - DXYEAB12, DXYEAB13, DXYEAB14, and DXYEAB15 furnish message help for the application's error messages.
- Certain file customization examples in the QMF1210.SDSQSLBE library:
  - DSQABB1J provides sample JCL for the batch job. It is a sample procedure that runs a query in batch mode.
  - DSQABB1P and DSQABB1S are sample QMF procedures. They provide an example of a procedure that submits the JCL for the job.

### Possible changes to the application

You can make the following changes to the application:

- Allow users to choose the Db2 subsystem.

Within the DSQABB1J sample file is the ISPSTART statement to call batch-mode QMF. This statement does not provide a value for the DSQSSUBS parameter of QMF. As a result, the Db2 subsystem under which QMF is to run is assumed to have the default name DSN. If you want QMF to run in a Db2 subsystem with a different name, add DSQSSUBS=*ssid* to the PARM operand of the ISPSTART command (where *ssid* is the appropriate subsystem name).

- Allow the user to specify a GDDM nickname for the printed report.
- Add extra logic to enforce your site's rules.

For example, you might offer the user a list of acceptable volumes when the user creates a new data set for the report output.

- Change the JCL produced by the application to conform to your site's needs.

You can do either of the following:

- Add accounting information to the JOB statement.
- Change the name of QMF application plan in the ISPSTART statement of the SYSTSIN data set. The default QMF plan name for Version 12.1 is QMF12.

You might also have to make additional changes, such as:

- Adding a field or fields to the prompt panel (DXYEABMP)
- Changing the help panels for the prompt panel
- Adding new error messages to DSQBE00, DSQBE01, or DSQBE02
- Changing some of the logic in DSQABB12

**Important:** Users who call the batch application should not maintain a data set named *userid*.DSQ1EBFT.PROC, where *userid* is the user's TSO logon ID. If such a data set exists, the QMF batch application might not run correctly.

### Example of modifying the application

The following example shows one way you can modify the BATCH application.

This example modifies the batch application so that all users have the same PROFILE PREFIX statement, and assumes that all users have unique user IDs. Add the user IDs to the data set names using &SYSUID and &ZUSER.

You must make three modifications to the DSQABB1S SKELETON file as shown here. The old lines are commented out. The new replacement lines follow them.

```

)CM -----
)CM FILE: DSQABB1S
)CM DESCRIPTION: THIS SKELETON CREATES DSQABB1S, THE PROC WHICH
)CM SAVES THE CURRENT FORM (IF SPECIFIED)
)CM IMPORTS AND SAVES THE PROC WHICH RUNS THE QUERY
)CM SENDS THE QMF INVOCATION JOB TO z/OS BATCH
)CM RESETS THE PROC ITEM
)CM FREES ISPF FILE USED FOR FILE TAILORING
)CM DISPLAYS THE QUERY PANEL
)CM -----

)SEL &FAN = &YES
&SAVE &FORM &AS &FNAME (&SHARE=&YES, &CONFIRM=&NO)
)ENDSEL

)CM &IMPORT &PROC &FROM '&ZPREFIX..DSQ1EBFT.&PROC.' (&MEMBER = DSQABB1P
&IMPORT &PROC &FROM '&ZPREFIX..&ZUSER..DSQ1EBFT.&PROC.' (&MEMBER = DSQABB1P
&SAVE &PROC &AS &PNAME (&CONFIRM=&NO)
)CM TSO SUBMIT '&ZPREFIX..DSQ1EBFT.&PROC.(DSQABB1J)'
TSO SUBMIT '&ZPREFIX..&ZUSER..DSQ1EBFT.&PROC.(DSQABB1J)'

TSO FREE FILE(ISPF FILE) DELETE
&RESET &PROC
)CM &IMPORT &PROC &FROM DSQABB
&IMPORT &PROC &FROM &ZUSER..DSQABB

)SEL &ITM = &QUERY
&DISPLAY &QUERY
)ENDSEL

```

Figure 113. Modifying the DSQABB1S SKELETON file

Make the five modifications to DSQABB12 CLIST as commented here.

```

/*****/ 00088000
/* ALLOCATE USERID.DSQ1EBFT.PROC TO BE USED FOR ISPF */ 00089000
/* FILE TAILORING OUTPUT. */ 00090000
/*****/ 00091000
FREE FILE(ISPF FILE) 00092000
/* ALLOC DDNAME(ISPF FILE) DSNNAME(DSQ1EBFT.&PROC) OLD 00093000
ALLOC DDNAME(ISPF FILE) DSNNAME(&SYSUID..DSQ1EBFT.&PROC) OLD 00093000
IF &LASTCC -= 0 THEN + 00094000
DO 00095000
FREE ATTRLIST(ATTRPDS) 00096000
ATTR ATTRPDS LRECL(80) RECFM(F B) BLKSIZE(800) DSORG(PO) 00097000
/* ALLOC DDNAME(ISPF FILE) DSNNAME(DSQ1EBFT.&PROC) NEW SPACE(5,2) + 00098000
/* TRACKS DIR(10) USING(ATTRPDS) CATALOG 00099000
ALLOC DDNAME(ISPF FILE) DSNNAME(&SYSUID..DSQ1EBFT.&PROC) NEW + 00098000
SPACE(5,2) TRACKS DIR(10) USING(ATTRPDS) CATALOG 00099000
END 00100000
IF &RC = 8 THEN + 00101000
DO 00102000
:
/*****/ 00203000
/* EXPORT CURRENT CONTENTS OF PROC PANEL */ 00204000
/*****/ 00205000
ISPEXEC SELECT PGM(DSQCCI) + 00206000
/* PARM( &EXPORT &PROC &TO DSQABB (&CONFIRM = &NO ) 00207000
PARM( &EXPORT &PROC &TO &SYSUID..DSQABB (&CONFIRM = &NO ) 00207000
IF &LASTCC -= 0 THEN DO 00208000
ISPEXEC SELECT PGM(DSQCCI) + 00209000
PARM(SET GLOBAL (DSQEC_NLFCMD_LANG = &LOCLANG )) 00210000
SET &MSG = &DSQB.023 00211000
ISPEXEC SELECT PGM(DSQCCI) PARM( &MESSAGE &MSG ) 00212000
SET &RCDE = 8 00213000
GOTO CLEANUP 00214000
END 00215000
:
/*****/ 00244000
/* IMPORT AND RUN FILE TAILORED SKELETON */ 00245000
/*****/ 00246000
ISPEXEC SELECT PGM(DSQCCI) + 00247000
/* PARM( &IMPORT &PROC &FROM DSQ1EBFT (&MEMBER = DSQABB1S ) 00248000
PARM( &IMPORT &PROC &FROM &SYSUID..DSQ1EBFT (&MEMBER = DSQABB1S ) 00248000
IF &LASTCC -= 0 THEN + 00249000
:

```



```

CLEANUP: FREE FILE(ISPF) DELETE 00274000
DONE: SET &ZPLACE = &SAVEPLC 00275000
      SET &ZPFCTL = &SAVEPFC 00276000
      SET &ZPF01 = &STR(&SAVEPF01) 00277000
      SET &ZPF13 = &STR(&SAVEPF13) 00278000
      SET &ZPF03 = &STR(&SAVEPF03) 00279000
      SET &ZPF15 = &STR(&SAVEPF15) 00280000
      SET &ZPF10 = &STR(&SAVEPF10) 00281000
      SET &ZPF22 = &STR(&SAVEPF22) 00282000
      SET &ZPF11 = &STR(&SAVEPF11) 00283000
      SET &ZPF23 = &STR(&SAVEPF23) 00284000
      ISPEXEC VPUT (ZPLACE ZPFCTL ZPF01 ZPF13) PROFILE 00285000
      ISPEXEC VPUT (ZPF03 ZPF15 ZPF10 ZPF22 ZPF11 ZPF23) PROFILE 00286000
/* DELETE DSQABB.&PROC 00287000
   DELETE &SYSUID..DSQABB.&PROC 00287000
   EXIT CODE(&RCDE) 00288000

```

Figure 114. Modifying DSQABB12 CLIST

## Initiating a QMF batch job in the foreground on ISPF or TSO

To start QMF in batch mode in the foreground, you can use any of the usual methods to start QMF.

For example, from TSO READY mode, you can issue the following statement to start QMF from a CLIST:

```
ISPSTART CMD(clist_name) NEWAPPL
```

In this statement, *clist\_name* is the name of the CLIST that starts QMF. This CLIST must contain a statement of the following form:

```
ISPEXEC SELECT PGM(DSQMFE) NEWAPPL(DSQE)
              PARM(...DSQSMODE=B,DSQSRUN=auth_ID.proc_name)
```

Here, the ISPSTART statement runs in the foreground, not in the background. You cannot do other things with TSO while waiting for the CLIST to end.

When the CLIST ends, you are back in TSO READY mode. Before the CLIST ends, you might see a display of the **ISPF Disposition Prompt** panel if your procedure terminates before you specify permanent disposition parameters for the TSO console, log, and list files. To avoid displaying this panel, specify permanent disposition parameters for these files. A value of D (specifying "delete") for each is probably adequate. If you do not know how to specify these dispositions, ask an ISPF expert or use ISPF help.

### Related concepts

#### [Starting QMF](#)

QMF can be started only from z/OS. QMF can be set up to run under TSO, ISPF, as a batch job, or under CICS.

## Debugging a batch-mode procedure or application

You can use the trace codes and the HELP command to diagnose problems with a batch-mode procedure.

L2 tracing, which traces messages and commands at the most detailed level, is the default for procedures that run in batch mode. To change the trace setting, add a SET PROFILE command to your procedure. For example, to specify L1 tracing instead of L2, add the following statement at the start of the procedure:

```
SET PROFILE (TRACE=L1
```

With either L1 or L2 tracing, a log is produced in the trace data set (which is DSQDEBUG by default). Within this log is a series of message records: one for each message that QMF issued while the procedure was being run.

With L2 tracing in effect, the log also contains a record for each QMF command run by the procedure (and its subordinates).

If the procedure terminates prematurely, an error message is written to the trace data. You can then use the HELP command to display the corresponding message help panel.

## Starting a QMF batch job from a remote Db2 client

Any Db2 client that is connected to the Db2 database where the QMF stored procedure DSQQMFSP is installed can start a QMF for TSO batch job. The batch job is started from the z/OS system that runs the QMF stored procedure.

### Before you begin

Before you start a batch job from a remote Db2 client, authorizations for the QMF stored procedure interface and batch mode must be set up. Also, see the following information before you begin:

- [“Installing the QMF stored procedure interface \(TSO only\)” on page 107](#)
- [“Authority to operate in batch mode” on page 319](#)
- [“JCL to execute a QMF batch job” on page 320](#)
- [“Debugging a batch-mode procedure or application” on page 331](#)

### Procedure

Issue an SQL CALL statement that calls the stored procedure to start the z/OS batch job. Include the name of the QMF procedure that is run by the QMF stored procedure as the first parameter on the CALL statement.

For example:

```
CALL Q.DSQQMFSP('W397754.LAUNCH_STAFF_REPORT','L2','','','')
```

### Example

The following example QMF REXX procedure (W397754.LAUNCH\_STAFF\_REPORT) starts a staff report batch job:

```
/* REXX - Launch Staff Report Batch Job. */
/* This procedure uses the REXX FtpApi to launch the batch */
/* job. */
/*****

/* Initialize REXX program variables */
hostname = "STLMVS1" /* z/OS host name to receive the batch job */
uid = "W397754" /* z/OS user id that is used to login to ftp */
uidpw = "xxxxxxx" /* user id pw used to login to ftp */
pcode = 0 /* Initialize return code */
TRACEID = 'PAZ' /* Set FTP trace flag */

/* Create REXX FTP environment */
ftp_rc = ftpapi('fcai.', 'create', TRACEID)
if ftp_rc < 0 then
do
pcode = -10
exit pcode
end

/* Connect to host. See hostname variable for name of host */
ftp_rc = FtpApi('fcai.', 'init', '-w 300 ||hostname)
if ftp_rc < 0 then exit -21

/* Logon user to FTP session */
cmd = "USER " uid
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -22

cmd = "PASS " uidpw
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -23

/* Set JES to receive JCL file */
cmd = "QUOTE SITE FILETYPE=JES"
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -24
```

```

/* Send JCL file to JES for execution */
cmd = "PUT 'W397754.STAFF.REPORT.JCL'"
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -25

/* Quit the FTP session */
cmd = "QUIT"
ftp_rc = FtpApi('fcai.', 'scmd', cmd, 'W')
if ftp_rc < 0 then exit -26

/* Terminate te REXX FTP Environment */
ftp_rc = ftpapi('fcai.', 'term')
if ftp_rc < 0 then exit -27

exit pcode

```

## Running QMF as a batch transaction on CICS

QMF batch transactions can be run from a terminal or as a transaction running without a terminal.

All resources needed by QMF are available throughout the user session. Run QMF procedures that can be used to generate a report in order to conserve resources. The procedures can be run non-interactively.

QMF runs interactively as a conversational transaction in CICS.

### Running batch activities from a terminal on CICS

You can run QMF from a terminal to produce a report.

For example, you can run the procedure that is shown here to produce a report that is located in CICS auxiliary storage. (QMF treats lines that begin with "--" as comments in QMF linear procedures.)

```

-- Procedure name: STATRPT1_PROC
--
-- Example QMF procedure to create an auxiliary CICS
-- temporary storage queue named STATRPT1
--
  RUN  QUERY STATRPT1_QUERY (FORM=STATRPT1_FORM)
  PRINT REPORT (QUEUE=STATRPT1,QUEUETYPE=TS)
--
-- End of procedure

```

*Figure 115. Producing a report in CICS auxiliary storage*

Execute the QMF transaction described here to run this procedure in batch mode:

```
QMFE M=B,I=STATRPT1_PROC
```

QMF runs this transaction without displaying any screens. Upon successful completion of the procedure, the report is in CICS storage queue STATRPT1. You can then view the report by using the transaction CEBR supplied with CICS:

```
CEBR STATRPT1
```

### Running batch activities without a terminal on CICS

Use the EXEC CICS START command to run a procedure while the terminal that is running a batch job is locked.

A terminal used to run a batch job is locked until QMF completes the transaction. The following example runs the QMF procedure STATRPT1\_PROC in batch mode without a terminal:

```
EXEC CICS START TRANSID(QMFE) FROM(M=B,I=STATRPT1_PROC)
```

When this transaction completes, the CICS storage queue STATRPT1 can be browsed using the transaction CEBR supplied with CICS.

## Debugging a procedure on CICS

QMF provides a facility that traces messages, commands, and QMF functions. This facility is useful when there is a problem running a QMF procedure in batch mode.

Messages and commands are traced automatically when QMF is running in batch mode. You can route this message trace to CICS auxiliary temporary storage or a transient data queue. For example, to run the STATRPT1\_PROC procedure and send the command and message trace to a CICS auxiliary storage queue with the name QMFMSG, issue a CICS START command similar to the following:

```
EXEC CICS START TRANSID(QMFE)
      FROM(M=B,I=STATRPT1_PROC,DSQSDBQN=QMFMSG,DSQSDBQT=TS)
```

Multiple QMF transactions can issue messages to the same trace area. QMF issues a CICS ENQ command on the queue name while it writes a trace entry. Each entry is marked with the terminal ID and task ID of the QMF transaction that created the trace entry.

If you want to route QMF trace output to CICS auxiliary storage, do not set full component-level tracing because temporary storage will fill up quickly. For anything other than message-level tracing, use a transient data queue to hold the trace output.

### Termination return codes

Termination return codes for QMF are:

**0**

Normal termination

**8**

Abnormal termination

### Related concepts

[Getting the right level of detail in your trace output](#)

You can trace all QMF functions in detail or trace individual QMF functions.

# Chapter 19. Troubleshooting and diagnosing problems

To isolate and resolve problems that your users have while using QMF, you can use the troubleshooting and support information.

## Applying QMF service

Whenever you experience a problem with QMF, first ensure that the service level of your QMF installation is current.

Determine your current level of service, and search the [IBM Software Support website](#) for the latest PTFs for QMF and its prerequisite products. Additionally, request QMF's preventative service planning (PSP) bucket. The PSP bucket contains general hints, APARs, and documentation changes associated with known problems. The PSP upgrade name for QMF Version 12 Release 1 is QMFC10. Valid subset names for PSP upgrade QMFC10 are shown here.

*Table 75. QMF PSP subset names found in PSP upgrade QMFC10*

| <b>PSP subset name</b> | <b>Applies to</b>          |
|------------------------|----------------------------|
| HHPCC10                | QMF HPO                    |
| HQDC110                | QMF Data Service           |
| HSQCC10                | English (QMF base product) |
| JSQCC1Q                | QMF for z/OS OTC           |
| JSQCC1C                | QMF Classic Edition        |
| JSQCC1E                | QMF Enterprise Edition     |
| JSQCC51                | U/C English                |
| JSQCC55                | Danish                     |
| JSQCC5G                | Canadian French            |
| JSQCC56                | French                     |
| JSQCC57                | German                     |
| JSQCC58                | Italian                    |
| JSQCC59                | Japanese                   |
| JSQCC5A                | Korean                     |
| JSQCC5B                | Brazilian Portuguese       |
| JSQCC5C                | Spanish                    |
| JSQCC5D                | Swedish                    |
| JSQCC5E                | Swiss French               |
| JSQCC5F                | Swiss German               |
| JYQCC10                | QMF applications           |

### Related concepts

Installation prerequisites for requester (Db2 for z/OS) databases

Before you can install QMF on Db2 for z/OS databases that function as stand-alone or requester databases, you must fulfill the hardware and software requirements.

### Related tasks

Determining the QMF service level

Running an SMP/E report against the target or distribution zones is always the best way to determine the service level. However, you can also determine the QMF service level in other ways.

### Related information

Search for the latest PTFs for QMF and its prerequisite products.

## Correcting common problems

---

Check the list of common problems and possible solutions before you try a more in-depth diagnosis.

### Errors that can occur at initialization time

Errors that occur during QMF initialization or the connect process generally do not stop QMF. However, these types of errors are usually logged to the QMF trace data.

The location of the trace data varies depending on your environment:

- In TSO, ISPF, and native z/OS, the trace data is stored in DSQDEBUG.
- In CICS, the trace data is stored in a transient data queue named DSQD, unless you changed the type or name of the queue using the DSQSDBQT and DSQSDBQN program parameters when you issued the command to start the QMF session.

In addition to looking at the trace data, check the screen for messages. If no messages are displayed and you are trying to start QMF for TSO, issue the following TSO command and restart QMF: PROFILE MSGID WTPMSG. This command logs messages to the screen in TSO.

After reviewing the trace output and any messages issued to the screen, read the following topics to determine the problem. If none of these errors appears to be the problem, review additional diagnostic aids that can help you determine the problem and diagnose its cause.

### Related concepts

Using diagnosis aids

Try to diagnose the problem by using a variety of diagnostic aids.

### Related tasks

The trace facility

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

### Problems that can occur when QMF is not using current load modules

Unexpected behavior, as well as errors or warning messages when initializing QMF or connecting to a remote database, can happen when QMF initializes with modules from one or more prior releases.

### Procedure

Follow these steps to check whether QMF is initializing with the correct load modules:

1. Issue the command SHOW GLOBALS and check the value of the DSQAO\_QMF\_VER\_RLS variable. The value of this variable for QMF Version 12 Release 1 is QMFV12R1 . 0.

If the variable value is not what you expect, it is likely that, during the initialization process, QMF is accessing load modules from prior releases. Depending on how you allocate the SDSQLOAD library, the load modules that QMF is accessing during initialization might be in an unexpected data set, the linklist, or the LPA.

Even when the DSQAO\_QMF\_VER\_RLS variable has a value of QMFV12R1.0, the initialization process might be interrupted by the following error when the DSQUOPTS module is loaded: [“DSQ90579” on page 342](#).

2. Determine the level of load modules that QMF is accessing. How you complete this step depends on whether you are able to start QMF.

- If you are able to start QMF:
  - a. Start the program with a value of ALL for the DSQSDEBUG parameter.
  - b. After you run the trace, review any resulting trace output. QMF Version 12.1 modules are associated with the eyecatcher V12R1.00 in the QMF trace. For example:

```

-----
***** 09/12/22 10.25.52 ***** -----
DSQCPARM: Applied Service Level (1)
34347370 DSQCABND V12R1.00 09/09/16 13:36 WIM11328
343476C8 DSQCADJM V12R1.00 09/09/16 13:36
345F50C8 DSQCAVAR V12R1.00 09/09/16 13:36 WIM11266
3464D8D0 DSQCBDS V12R1.00 09/09/16 13:36 WIM7688
34347E88 DSQCBMPA V12R1.00 09/09/16 13:36
34349290 DSQCBOR V12R1.00 09/09/16 13:36 WIM11215
346B69D8 DSQCB64C V12R1.00 09/09/16 13:36 WIM11525

```

- If you are unable to start QMF:
 

Inspect any resulting dump output (such as SYSUDUMP, DSQUDUMP, or SYSMDUMP output) for the version and release information. Verify that the eyecatcher that is associated with all the QMF modules is the same. QMF Version 12.1 modules are associated with the eyecatcher V12R1.00, as shown in the preceding example.
3. If you find that the QMF initialization process is using load modules from one or more prior releases instead of the current release, determine the location from which the load modules are being accessed.

- If you are running QMF for TSO under ISPF, you can use the ISPF ISRDDN diagnostic utility to obtain information about file allocations for the QMF load modules. ISRDDN is a utility that provides a list of allocated ddnames within a TSO user's address space. Thus, how you use this facility to view QMF file allocations depends on how these allocations are done. For example, if your QMF file allocations are done through ISPF, you can issue the following command from within QMF: TSO ISRDDN. The command returns a list of current data set allocations. Browse the list and check that all data sets are being accessed from the correct locations. For sample allocations for the data sets, see the following information:

- [“Customizing requester installations under TSO” on page 49](#)
- [“Customizing requester installations under CICS” on page 55](#)

The default data set prefix for QMF Version 12.1 data sets is QMF1210. Your data set prefix for the QMF Version 12.1 load modules might be different.

If you find allocations that are not pointing to the QMF Version 12.1 libraries, change these allocations to point to the correct locations.

- You can also use the ISRDDN utility to retrieve allocation information for specific modules and optionally browse the modules if necessary. For example, you can issue the following command to determine the location from which the DSQQMF module was loaded and optionally browse the module: TSO ISRDDN MEMBER DSQQMF

Be sure that the DSQQMF module is being accessed from the expected location and that it contains the version and release identifiers that you expect. For example, the following output shows that the DSQQMF module was loaded from the QMF.COM.QMF1210.SDSQLOAD data set, which was accessed via ddname DSQLLIB. The specific name of your QMF data sets will vary.

If the QMF Version 12.1 data set prefix you chose does not include the version and release information, use the following command to browse the DSQQMF load module to help you determine this information: TSO ISRDDN BROWSE DSQQMF

For QMF Version 12.1, verify that the eyecatcher matches V12R1.00, as shown in the following figure. The date, time, and service level of the module can vary depending on which PTFs were applied.

```

BROWSE      DSQQMF JPA Start:34484B48 Size:0038BE10      Line 00000000 Col 001 080
***** Top of Data *****
+0 (34484B48) 47F0F122 2EC4E2D8 C3C9D5E3 4040E5F1 *          DSQCINT V1 *
+10 (34484B58) F0D9F14B F0F04040 404040F0 F961F0F9 * 1R1.00 12/12 *
+20 (34484B68) 61F1F640 F1F37AF3 F740E6C9 D4F1F1F5 * /16 13:37 WIM125 *
+30 (34484B78) F9F840D3 89838595 A2858440 D481A385 * 33 Licensed Mate *
+40 (34484B88) 99898193 A2406040 D7999697 8599A3A8 * rials - Property *
+50 (34484B98) 40968640 C9C2D440 F5F6F3F5 60C4C2F2 * of IBM 5615-DB2 *
+60 (34484BA8) 6B40F5F6 F0F560C4 C2F2404D C35D40C3 * , 5697-P43 (C) C *
+70 (34484BB8) 9697A899 898788A3 40C9C2D4 40C39699 * opyright IBM Cor *
+80 (34484BC8) 974B40F1 F9F8F26B 40F2F0F1 F04B40C1 * p. 1982, 2013. A *
+90 (34484BD8) 939340D9 898788A3 A240D985 A28599A5 * ll Rights Reserv *
+A0 (34484BE8) 85844B40 E4E240C7 96A58599 95948595 * ed. US Governmen *
+B0 (34484BF8) A340E4A2 8599A240 D985A2A3 998983A3 * t Users Restrict *
+C0 (34484C08) 858440D9 898788A3 A2406040 E4A2856B * ed Rights - Use, *
+D0 (34484C18) 4084A497 93898381 A3899695 40969940 * duplication or *
+E0 (34484C28) 8489A283 9396A2A4 99854099 85A2A399 * disclosure restr *
+F0 (34484C38) 8983A385 844082A8 40C7E2C1 40C1C4D7 * icted by GSA ADP *
+100 (34484C48) 40E28388 8584A493 8540C396 95A39981 * Schedule Contra *
+110 (34484C58) 83A340A6 89A38840 C9C2D440 C3969997 * ct with IBM Corp *
+120 (34484C68) 4B4090EC D00C187F 41807FFF 41908FFF * -----*
Command ==>                               Scroll ==> PAGE

```

Figure 116. Using the ISRDDN utility to browse the DSQQMF module to obtain version and release information

### Related tasks

#### The trace facility

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

#### Abends

Certain abends can occur at initialization time. You might need to use native z/OS, TSO, or CICS diagnostic facilities to help you diagnose an abend.

For more information about common abends at initialization, see one of the following sections:

- [“0C1” on page 338](#)
- [“0C4” on page 338](#)
- [“0C7” on page 339](#)
- [“ASRA” on page 339](#)
- [“AEY9” on page 340](#)
- [“AZTS” on page 340](#)
- [“G050” on page 340](#)

#### 0C1

If you experience an 0C1 abend, the problem could be that the QMF initialization process is using load modules from one or more prior releases instead of the current release. See [“Problems that can occur when QMF is not using current load modules” on page 336](#) to determine whether this is situation is the problem.

#### 0C4

If you experience an 0C4 abend in response to a QMF command that would normally display a panel, take the following steps to troubleshoot the problem:



1. Check to ensure that you ran the DSQ1nPNL installation job, where *n* is a 1-character national language identifier. This job populates the VSAM panel library, DSQPNLn, with the product panels and help panels from QMF Version 12.1. For more information about this job, see [“Populating the VSAM panel library”](#) on page 37 or the QMF program directory. Be sure that you are using the correct program directory for your QMF release and national language.
2. If you are still experiencing the problem after the DSQ1nPNL job ran successfully, check file allocations on the following DD statements:
  - QMF for TSO:  
Check your allocation for DD statement ADMGGMAP to make sure that it points to QMF1210.SDSQMAPn.
  - QMF for CICS:  
Check that the ADMF DD statement in the startup JCL for the CICS region in which QMF resides correctly points to the GDDM ADMF data set. The DSQ1nADM installation job writes the QMF Version 12.1 GDDM maps to this data set.
3. If the file allocations are correct, see [“Problems that can occur when QMF is not using current load modules”](#) on page 336 for more troubleshooting suggestions. Abends can occur at various times when QMF Version 12.1 is mistakenly initializing with modules from one or more prior releases.

## OC7

If you experience an OC7 abend, the problem could be that the QMF initialization process is using load modules from one or more prior releases instead of the current release. See [“Problems that can occur when QMF is not using current load modules”](#) on page 336 to determine whether this situation is the problem.

## ASRA

ASRA abends can occur if you are running QMF for CICS. If you receive this error, check the following possible areas for the problem:

- On QMF startup:
  - Make sure that all GDDM link-edits ran successfully.
  - Make sure that the GDDM IVPs ran successfully in the CICS region.
  - Verify that the QMF SMP/E APPLY steps that are documented in the QMF program directory completed successfully. These steps link the QMF load modules with the CICS load modules. Be sure that you are using the correct program directory for your QMF release and national language.
  - Make sure that the region allocates the QMF Version 12 Release 1 load libraries and map groups. For more information about these steps of the installation and customization process, see one of the following topics:
    - [“Updating the CICS startup job stream”](#) on page 59
    - [“Loading QMF GDDM maps to the GDDM ADMF data set”](#) on page 58
- In module DSQQMFE, CSECT ADM:
 

The problem is probably a GDDM failure. Verify that GDDM is correctly installed and customized for CICS. Verify that GDDM is in the same CSI zone as CICS.
- In module DSQQMFE, CSECT DSQEGINT:
 

Verify that GDDM is customized for CICS and that a CSD entry exists for the GDDM module ADMASPLC.
- In module DSQQMFE, CSECT DSQIELI:
 

Verify that a CSD entry exists for the Db2 for z/OS interface module DSQIELI.
- In module DSQCBST, CSECT DSQCMCVP:

After QMF service is applied, verify that a z/OS LLA REFRESH was done in case QMF code is in the Library Lookaside.

- On exit of QMF:

Check that the governor is linked correctly. Review job DSQ1EGLK.

- With ABEND0C4 and DFHSM0102:

This error occurs when you run a query or when you press the Help function key. Make sure that the CSD file entry for DSQPNL $n$  has RECFM=V (where  $n$  is a 1-character national language identifier).

- On issuing HELP or RUN commands:

QMF data set DSQPNL $n$ , which contains help and other screen text, was either not installed correctly or it was not allocated to the job that started the CICS region. To correct the problem:

- Verify that the CSD entry is defined correctly.
- Verify that a DD statement for DSQPNL $n$  exists in the job stream that starts the CICS region. DD statements are described in [“Updating the CICS startup job stream”](#) on page 59.

Also, look for console error messages that are related to the DSQPNL $n$  data set.

Replace  $n$  with the 1-character language identifier that represents the NLF you are using.

### **AEY9**

The Db2 for z/OS attachment facility is not active in the CICS region. Start the attachment facility by using the DSNB transaction.

### **AZTS**

Make sure that GDDM is running with IOSYNCH=YES.

### **G050**

Verify that the release level of GDDM that you customized for CICS matches the release level of GDDM that you are using in the job stream to start the CICS region.

### **Related concepts**

[Using diagnostics native to the environment](#)

You might need to diagnose abends using diagnostic facilities in TSO, z/OS, or CICS.

### **Related tasks**

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### **Related information**

See the information about QMF Version 12.1 program directories.

### **QMF messages**

If you experience an error during QMF initialization, you might recover by checking for error messages and taking appropriate action.

### **DSQI0026**

This message usually occurs on startup. If you are running QMF under CICS, make sure that the QMF $n$  transaction is entered from a clear screen.

### **DSQ10297**

Invalid subsystem ID.

This error can occur on ISPF startup, when you use the callable interface, or when you start QMF for TSO as a Db2 for z/OS stored procedure. Check that the DSQSSUBS parameter (for which the short name is S) refers to the subsystem in which QMF is being started.

If you are running QMF as a Db2 for z/OS stored procedure, the DSQSSUBS parameter should refer to the subsystem in which Q.DSQQMFSP procedure is installed. For more information about starting QMF, see one of the following topics:

- [“Starting QMF with ISPF” on page 72](#)
- [“Installing the QMF stored procedure interface \(TSO only\)” on page 107](#) and [“Starting QMF as a Db2 for z/OS stored procedure” on page 119](#)
- , which provides information about starting QMF under the callable interface

### **DSQ10344**

A coded character set identifier (CCSID) contains all of the information necessary to assign and preserve the meaning and rendering of characters through various stages of processing and interchange. This information always includes at least one code page. The CCSID also has an associated encoding scheme that governs how various code points are to be handled.

At initialization time, QMF compares the GDDM application code page (specified by the APPCPG parameter) with two different CCSID values:

- CURRENT APPLICATION ENCODING SCHEME special register  
The application encoding scheme for the database is specified in this special register in Db2 for z/OS. If QMF is running in an SBCS language, the GDDM application code page should match the value of this special register.
- SYSIBM.SYSTEM\_EBCDIC\_CCSID session variable  
This session variable specifies the CCSIDs that are in use for EBCDIC data on Db2 for z/OS. If QMF is running in a DBCS language, the GDDM application code page should match the value of this session variable.

If the GDDM application code page does not match one of the preceding two database CCSIDs, message DSQ10344 is issued as a warning that there might be a difference between the data that is sent by Db2 to GDDM for display and what is displayed. To correct any potential display problems, you need to change the GDDM CCSID by modifying the APPCPG external default. For more information, see one of the following topics:

- [“Customizing GDDM external defaults” on page 49](#) for TSO
- [“Customizing GDDM external defaults” on page 57](#) for CICS

For more details, see the GDDM documentation in [the IBM Publications Center](#).

### **DSQ10493**

This message indicates a database authorization error. Verify that the Db2 for z/OS resource control table (RCT) contains an entry for the transaction ID that you are using to start QMF. For example, if you are using the CICS transaction ID QMFE to start QMF, code the entry as follows:

```
DSNCRCT TYPE=ENTRY ,TXID=QMFE ,PLAN=QMF12 ,AUTH=DEPT1
```

In this example, the authorization ID is DEPT1, and the plan ID is QMF12.

### **DSQ22843**

Make sure that GDDM is running with IOSYNCH=YES.

If the QMF IVP fails with the message `A GDDM graphics printer nickname is required for printer`, there is an error in your GDDM nicknames definition file.

The QMF IVP includes a step to print a query. If you use GDDM nicknames at your site, change the PRINT QUERY statement in the IVP procedure to PRINT QUERY (PRINTER = *gddmnickname*. The procedure for creating GDDM printer nicknames is provided in [Chapter 13, “Setting up printing and charting functions,”](#) on page 213.

### **DSQ90579**

This error indicates that you are trying to start QMF for TSO or QMF for CICS under Db2 QMF for z/OS, but the OTC\_LICENSE variable in the DSQUOPTS exit routine is set to TERMS\_ACCEPTED. To correct this problem:

1. Change the value of the OTC\_LICENSE variable in the DSQUOPTS exit routine to NOT\_USED.
2. Assemble and link-edit the routine as explained in [“Setting global variables with the DSQUOPTS routine”](#) on page 132.
3. Place the modified DSQUOPTS load module in an exit library that is allocated in the initialization procedure that starts QMF. When you use sample job DSQ1UOPT to assemble and link-edit the DSQUOPTS exit routine, the modified DSQUOPTS load module is placed in the default QMF exit library, QMF1210.SDSQEXIT.

For more information about the DSQUOPTS exit routine, see [“Initializing global variables and QMF session behavior when QMF starts”](#) on page 131.

### **DSQ50405**

The following error message when you start QMF can indicate that QMF is initializing with a load module library (SDSQLOAD) from a previous release:

```
Number of REXX program parameters returned should be &V1, not &V2.
```

Numbers are shown in the message in place of the variables. This problem can happen even when you correctly allocated the QMF Version 12.1 SDSQLOAD library if there are load modules from a previous release in the linklist or LPA. The message indicates that QMF is accessing a Version 12.1 copy of QMF1210.SDSQEXCE(DSQSCMDE), but a previous copy of the SDSQLOAD load library. See [“Problems that can occur when QMF is not using current load modules”](#) on page 336 for steps you can take to diagnose and correct the problem.

### **QMF error messages associated with SQL code -805**

The following QMF errors are associated with SQL code -805 and can occur during initialization:

- DSQ10205
- DSQ11205
- DSQ12105
- DSQ13005
- DSQ14152
- DSQ14153
- DSQ14154
- DSQ15805
- DSQ16805
- DSQ17805
- DSQ30805
- DSQ31805
- DSQ35805
- DSQ36805

Record all the tokens that are returned from the SQL code -805. For more information about this SQL code, see the Db2 documentation. Refer to the following page for troubleshooting tips: <http://www.ibm.com/support/docview.wss?uid=swg21567609>

### **QMF error messages associated with SQL code -551**

During initialization, QMF checks whether the authorization ID of the user who starts QMF has either the INSERT or DELETE privilege for the Q.PROFILES control table. If the authorization ID has either of these privileges, QMF considers that user to be a QMF administrator. If the user is not a QMF administrator, a -551 SQL code is issued for the INSERT privilege, the DELETE privilege, or both. This SQL code is expected under these circumstances. The message is not displayed to the end user, nor is it captured in the QMF trace data set (DSQDEBUG) or in the error log table (Q.ERROR\_LOG) because it is not an error. The message does not stop QMF initialization or cause any problems during QMF operation. However, if you need to suppress the -551 SQL code for any reason, you must disable QMF administrator authority checking according to the procedure in [“Initializing global variables and QMF session behavior when QMF starts” on page 131.](#)

The following QMF messages are associated with SQL code -551:

- DSQ10951
- DSQ11851
- DSQ12751
- DSQ15551
- DSQ16551
- DSQ17551
- DSQ30551
- DSQ31551
- DSQ35551
- DSQ36551

For more information about QMF administrator authority, see [“Required authorities for QMF administration” on page 13.](#)

### **Problems starting QMF**

Use a troubleshooting process for problems starting QMF to determine the cause of the problem and to find a resolution.

If you cannot start QMF, troubleshoot the problem in the following way:

- Determine whether the problem with starting QMF applies to all users or just a certain individual or group.
- Check whether there are any messages on the display screen, and look up the explanation for the message you see in the trace data in .
- If nothing appears on the screen and nothing is in the trace data, go into ISQL and issue a `SELECT * FROM Q.ERROR_LOG` statement and check whether any entries appear during the time you were trying to access QMF.
- QMF initializes Db2 and GDDM during QMF initialization. If any DSN (Db2) and ADM (GDDM) error messages appear, look them up in the messages and codes information for the appropriate product.

Check that the Db2 database is initialized and working properly. If all users are getting a type of ADMxxxx message when trying to start QMF, check that the base GDDM product is working correctly by running the GDDM IVPs.

- Check that the QMF initialization process is accessing load modules for the current release. Initialization problems can result if QMF is being initialized with modules from one or more prior releases. See [“Problems that can occur when QMF is not using current load modules” on page 336](#) for more information.

An error can occur when an incorrect version of the DSQUOPTS exit routine is being loaded at initialization time or the value of the OTC\_LICENSE variable is not appropriate for the edition of QMF that you are trying to start. See [“DSQ90579” on page 342](#) for more information.

### **Problems with command synonyms, function keys, and resource control tables**

When you have problems with command synonyms, function keys, and resource control tables, you might recover by reviewing the user's privileges or the table structure and data

For command synonyms, function keys, and resource control tables, ensure that:

- The user has the SQL SELECT privilege for the relevant table. If this might be the problem, issue an SQL GRANT statement.
- The table conforms to the proper structure:
  - The structure for command synonym tables is shown in [“Customizing Command synonyms” on page 234](#).
  - The structure for function key tables is shown in [“Customizing QMF function keys” on page 239](#).
- All rows of the table contain valid data. If this might be the problem, see:
  - [“Entering command synonym definitions into the table” on page 235](#) for information about valid command synonym definitions
  - [“Entering your function key definitions into the table” on page 246](#) for information about valid function key definitions
- All rows in the tables are unique.
- The SYNONYMS and PFKEYS fields of the user's QMF profile contain the name of the table where the command synonym definitions and function key definitions are stored. To see the names of the tables currently in use, see one of the following global variables:
  - The DSQAP\_SYNONYM\_TBL global variable contains the name of the command synonyms table in use for the current QMF session.
  - The DSQAP\_PFKEY\_TABLE global variable contains the name of the function key table currently in use.

### **Warning messages after you start QMF**

When a warning message is issued, the cause of the warning is written to the QMF trace data set, DSQDEBUG.

Warning messages after you start QMF might be caused by using the same authorization ID for both TSO and CICS. If a user has the same database authorization ID under both TSO and CICS and uses the same QMF profile for both environments, QMF might issue warnings. For example, you might see warnings about invalid entries for environment-specific commands in the Q.COMMAND\_SYNONYMS table. Though you can set up a QMF command synonym table that contains commands for both environments, users might receive errors if they issue commands that are not specific to the environment in which QMF is running at the time. To eliminate warning messages caused by this problem, allocate unique QMF profiles for each environment.

#### **Related concepts**

[Allocating the trace data set](#)

Make sure that the DSQDUMP data set or DSQDEBUG data set is allocated, depending on how you run QMF.

### **Incorrect output**

Sometimes you can tell that there is a problem without receiving an error message. The most common type of this error is incorrect output.

For example, the QMF home panel does not read Version 12 Release 1, but instead points to another release. If you are experiencing this error, correct it in one of the following ways:

- If you are using QMF for TSO, check your allocation for DD statement ADMGGMAP to make sure that it points to QMF1210.SDSQMAPn.
- If you are using QMF for CICS, check that the ADMF DD statement in the startup JCL for the CICS region in which QMF resides correctly points to the GDDM ADMF data set. The DSQ1nADM installation job writes the QMF Version 12.1 GDDM maps to this data set.

Unexpected behavior can also occur when QMF is using a mixed set of load modules from the current release and one or more prior releases.

**Note:** Incorrect output seen on displayed reports or printed reports can be caused by improperly ordered queries. If QMF does not have enough storage to complete the displayed and or printed report and has to make multiple trips to the data base to re-retrieve data, it is imperative that the report data be in the same order. Usage codes such as BREAK and GROUP require that data be in the same order for the processing of the report. This is achieved by specifying ORDER BY statements on the SQL query that is involved in the report. See the BREAK and GROUP edit codes for more information.

### Related tasks

Problems that can occur when QMF is not using current load modules

Unexpected behavior, as well as errors or warning messages when initializing QMF or connecting to a remote database, can happen when QMF initializes with modules from one or more prior releases.

## Problems with printing

The source of printing errors can either be in QMF or in GDDM.

### GDDM errors

If a GDDM error occurred during printing, QMF displays a message with the GDDM printer nickname.

QMF displays this message:

GDDM error using *nnnnnnnn*. See message help for details.

The character string *nnnnnnnn* in the message represents a GDDM printer nickname. Press the Help key to display the help panel, which contains an explanation of the error. This topic explains some common errors and what you can do to fix them.

GDDM printing errors begin with the characters ADM.

| GDDM error code | Message                                                                                                                    | Explanation and possible solution                                                                                                                                                                            |
|-----------------|----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DSQ50623</b> | GDDM error. ADM0307 E FILE 'ADMPRINT.REQU—QUEUE' NOT FOUND. Severity 8. Function DSOPEN. ***<br>CMD=PRINT                  | QMF cannot find a nickname definition for the printer name the user specified. You must set up a nickname definition for the printer name, or supply one that is already defined.                            |
| <b>DSQ50623</b> | GDDM error. ADM0314 E UNABLE TO OPEN 'MYPRINT'. DD STATEMENT MISSING. Severity 8. Function DSOPEN. ***<br>CMD=PRINT        | QMF was unable to find a DD statement for the output. You must provide a DD statement to your QMF startup EXEC, CLIST, or JCL to specify what to do with output from the nickname.                           |
| <b>DSQ50623</b> | GDDM error. ADM0482 E DEVICE NAME LIST '31E' IS INVALID FOR FAMILY 1. Severity 8. Function DSOPEN. ***<br>CMD=PRINT        | Your nickname definition is incorrect. The device token you supplied is not a valid token for the type of GDDM printer for which you created the nickname.                                                   |
| <b>DSQ50631</b> | GDDM error. ADM0904 E ALPHANUMERIC FIELDS ARE NOT SUPPORTED FOR THIS DEVICE. Severity 8. Function ASDFLD. ***<br>CMD=PRINT | The output the user is trying to print is not valid for the type of printer defined by the GDDM nickname. Certain types of output, such as QMF charts, are restricted to specific families of GDDM printers. |

Table 76. Common GDDM errors (continued)

| GDDM error code | Message                                                                                                             | Explanation and possible solution                                                                                                                                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DSQ90551</b> | GDDM error. ADM0055 E SPINIT, AT '82F810C2'X ADM0050 E DEFAULTS ERROR. INVALID SYNTAX OR VALUE AT '...JIP,ADMMNICK' | You might see a message like this when starting QMF. The message indicates that you made a syntax error somewhere in the ADMMNICK specification for the nickname. After you fix the syntax error, reload the ADMADFC GDDM external defaults module. |
| <b>DSQ50633</b> | GDDM error ADM0327 E 'TD WRITEQ' ERROR CODE '08000000'X, ON 'SYSP'. Severity 8. Function FSFRCE. *** CMD=PRINT      | A message like this indicates that the temporary storage or transient data queue (SYSP) to which QMF is attempting to print is closed, or that a DD statement is missing from your startup JCL.                                                     |

### Related information

Search for explanations about GDDM errors, about valid device tokens, and about output by family of printer in the GDDM documentation.

### QMF errors

If the source of a printing error is QMF, you can often recover.

The information in the following table helps you to solve errors that can occur during printing:

Table 77. QMF printing errors and possible solutions

| Symptom                                                                                                                                              | What it means                                                                                                                               | What to do                                                                                                                                                                                                                                                                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| You issue the PRINT command from the command line or a function key and see the following message:<br>GDDM printer nickname is required for PRINTER. | The object you are trying to print needs a printer name, and no default printer name exists in your profile.                                | Press the Enter key again to display a prompt panel on which you can enter a printer name and other print parameters. You can set a default printer name in your profile to avoid being prompted.                                                                                                                                                     |
| You issue several PRINT commands but find that only the last object is printed.                                                                      | Your output data set does not have a disposition of MOD, so each PRINT operation reopens the data set and overwrites the previous contents. | Change the disposition of your output data set to MOD. You cannot use the disposition MOD with a member of a regioned data set.                                                                                                                                                                                                                       |
| You print a QMF object and see unexpected control characters in the printed output or data set.                                                      | The device token or PROCOPT you are using does not match the device on which you are actually printing.                                     | Supply the correct device token, or reduce control characters to a minimum by one of these techniques: <ul style="list-style-type: none"> <li>For a report, table, procedure, profile, or a SQL or QBE query, specify PRINTER=' ' to bypass GDDM printing.</li> <li>For other objects, use PROCOPT=( (PRINTCTL, 0) ) with no device token.</li> </ul> |
| When printing a report, table, procedure, profile, or a SQL or QBE query, you see the following message:<br>File DSQPRINT did not open.              | No default printer name exists in your profile, and no DSQPRINT data set or system output is currently allocated.                           | Allocate DSQPRINT before issuing a PRINT command.                                                                                                                                                                                                                                                                                                     |



If you are using QMF under TSO and you allocate output from DSQPRINT to go to the HOLD queue, to release the output to the OUTPUT queue you must issue the following TSO command: FREE DDNAME (DSQPRINT)

## Display errors

If a report has several display control characters in it, data in one or more of the table columns from which the report is derived might be binary (rather than character) data. You can handle these control characters with the HEX function or with edit codes.

### Using the HEX function

The HEX function is an SQL scalar function that converts its argument to a string of legitimate characters. The resulting string is the value of the argument in hexadecimal notation. For example, the function argument ABC produces the string C1C2C3 in hexadecimal notation.

Instruct users to use the word HEX in their queries in front of any columns that might contain binary data. For example, the following statement converts binary data in column A of the table SMITH.TABLEA.

```
SELECT HEX(A) FROM SMITH.TABLEA
```

### Using edit codes for hexadecimal and binary data

The edit codes provided with QMF allow QMF to display binary data in character columns: X or XW (for hexadecimal data), and B, BW, C, or CW (for binary data).

### Handling binary data with user-written edit routines

Using the HEX function or the HEX and bit edit codes can be a good way to handle binary data. You can create your own edit code and write an edit exit routine in COBOL, PL/I, or Assembler to convert the binary data to the character string you want. You might consider predefining some QMF forms for users that use the new edit codes that you create.

### Related concepts

[Custom edit exit routines for QMF forms](#)

QMF forms help users to control the format of data returned from the database. If the default edit codes do not meet the report-editing needs of your site, you can create your own edit codes.

## Resolving storage-related issues

If you receive the following error messages: DSQ14352; DSQ14362; DSQ14484; or DSQ14504 (Not enough above-the-bar storage exists to complete the request), see [“Above the bar storage requirement changes” on page 88](#). Note that starting in QMF Version 11, QMF makes use of above the bar storage for XML and LOB data types.

## Managing QMF performance

---

You can monitor the performance of QMF and find some solutions or workarounds to common performance issues.

### Capturing EXPLAIN information for dynamic statements

You can monitor the performance of eligible dynamic SQL queries by capturing EXPLAIN information for those queries. EXPLAIN data contains information about the access paths that are used to process SQL statements.

#### Before you begin

The following prerequisites are required:

- Db2 for z/OS is in Version 10 new-function mode or later

- The PLAN\_TABLE and DSN\_STATEMENT\_CACHE\_TABLE tables exist on the Db2 for z/OS server

### About this task

The behavior of the EXPLAIN facility for dynamic SQL statements is controlled by the CURRENT EXPLAIN MODE special register. In QMF, you can set this special register through the DSQEC\_EXPL\_MODE global variable. This global variable affects only those statements that are issued through the RUN QUERY command.

### Procedure

To capture EXPLAIN information for dynamic SQL statements:

- Set the DSQEC\_EXPL\_MODE global variable to YES or EXPLAIN.

A value of YES causes explainable dynamic SQL statements to run normally. Information is captured in EXPLAIN tables after each statement is prepared and executed.

A value of EXPLAIN causes explainable dynamic SQL statements to not execute, but information is captured to EXPLAIN tables after each statement is prepared in the application. Applications that depend on the actual successful execution of statements fail if they run while this global variable is set to EXPLAIN. Only applications with simple application logic should use this option.

### Related concepts

#### Solving storage problems

Users might notice slow performance in running queries or formatting reports when not enough virtual storage is available to retrieve all rows required by the operation. You can recover from this problem by increasing storage in one or more areas.

#### Solving resource contention problems

SELECT queries can time out when required rows are being updated by other operations. To increase concurrency of your applications and limit resource contention on Db2 for z/OS, you can set the QMF DSQEC\_CON\_ACC\_RES global variable according to your site's needs.

#### Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement

When Db2 dynamic statement caching is active, you can specify that Db2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This can result in more sharing and reuse of cached statements.

### Related tasks

#### Enabling QMF queries to be eligible for query acceleration

Using a query accelerator that is available to the Db2 for z/OS database can significantly improve the performance of certain kinds of queries. QMF queries can be eligible to run on a query accelerator if they meet the prerequisites and conditions for query acceleration that are listed in the Db2 documentation.

#### Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting

Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

## Enabling QMF queries to be eligible for query acceleration

Using a query accelerator that is available to the Db2 for z/OS database can significantly improve the performance of certain kinds of queries. QMF queries can be eligible to run on a query accelerator if they meet the prerequisites and conditions for query acceleration that are listed in the Db2 documentation.

### About this task

In general, QMF queries inherently meet the eligibility requirements for query acceleration. For example, the QMF RUN QUERY command runs user-issued SQL statements dynamically. However, you must complete the following procedure to ensure that QMF queries are eligible for query acceleration.

### Procedure

To enable QMF queries to be eligible for query acceleration:

- Ensure that the CURRENT QUERY ACCELERATION special register is set to the appropriate value for query acceleration.

You can determine the value of the CURRENT QUERY ACCELERATION special register by running the following query:

```
SELECT CURRENT QUERY ACCELERATION
FROM SYSIBM.SYSDUMMY1
```

The initial value of the CURRENT QUERY ACCELERATION special register is determined by the value of the Db2 QUERY\_ACCELERATION subsystem parameter. The SET CURRENT QUERY ACCELERATION statement can be issued in a QMF SQL query to override the current value of the QUERY\_ACCELERATION subsystem parameter.

- Depending on the capabilities of your query accelerator, the QMF DSQSMRFI (MR) program parameter might be of importance. Some query accelerators do not support queries that are run with rowset cursors. If QMF is started with the DSQSMRFI (MR) program parameter set to YES, QMF uses a rowset cursor.

For some query accelerators, using a rowset cursor makes queries ineligible for query acceleration. To check the current value of the DSQSMRFI (MR) parameter, issue the SHOW GLOBALS command and check the value of the DSQAO\_DSQSMRFI variable. A value of 0 for the DSQAO\_DSQSMRFI global variable means that NO was specified for the DSQSMRFI program parameter. A value of 1 for the DSQAO\_DSQSMRFI global variable means that YES was specified for the DSQSMRFI program parameter.

### Related concepts

#### Solving storage problems

Users might notice slow performance in running queries or formatting reports when not enough virtual storage is available to retrieve all rows required by the operation. You can recover from this problem by increasing storage in one or more areas.

#### Solving resource contention problems

SELECT queries can time out when required rows are being updated by other operations. To increase concurrency of your applications and limit resource contention on Db2 for z/OS, you can set the QMF DSQEC\_CON\_ACC\_RES global variable according to your site's needs.

#### Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement

When Db2 dynamic statement caching is active, you can specify that Db2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This can result in more sharing and reuse of cached statements.

### Related tasks

#### Capturing EXPLAIN information for dynamic statements

You can monitor the performance of eligible dynamic SQL queries by capturing EXPLAIN information for those queries. EXPLAIN data contains information about the access paths that are used to process SQL statements.

#### Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting

Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

## Solving storage problems

Users might notice slow performance in running queries or formatting reports when not enough virtual storage is available to retrieve all rows required by the operation. You can recover from this problem by increasing storage in one or more areas.

### Increasing the user's report storage

If you are seeing storage-related errors or performance problems, you can make adjustments to users' virtual storage or spill data configurations.

- Adjusting virtual storage for reports

Users might experience slow performance if they do not have enough virtual storage to accommodate large reports. For example, if you set the DSQSBSTG parameter at a very low value and the user runs a query that retrieves hundreds of thousands of rows, QMF can only maintain a small amount of data in memory. The user might find performance slow for formatting complex reports or scrolling the report.

To maximize report performance, make sure that you specify an adequate amount of virtual storage for the user, using the DSQSBSTG or DSQSRSTG parameter. To provide the best performance, use a value that accommodates the largest report the user is likely to require. However, be aware that setting the DSQSRSTG parameter at a very high value can cause slow performance.

- **Defining spill storage for reports**

In QMF for TSO, you can specify the use of extended virtual storage to store data no longer needed in active storage, thus improving report performance.

You can also spill data to a file in both QMF for TSO and CICS. However, using primarily virtual storage for QMF operations provides better performance. Users who rely on a spill file and have little virtual storage might notice slow performance for large reports. Even with a spill file, a user can encounter an incomplete data condition if the spill file is not large enough to hold all the data. For CICS, because a spill file can hold a maximum of 32,767 rows of size 4 KB each, setting DSQSBSTG higher ensures that QMF will complete the report.

If you are using a spill file, performance might also slow down if QMF needs a data row (as a result of a SCROLL BACKWARD command) and that data is not in the spill file or in virtual storage. In this case, an I/O abend occurs. QMF provides error-handling routines for the DCB SYNAD exit and recovers from these I/O abends using information provided using the DCB abend exit. QMF then quits using the spill file and fetches the data again from the database.

**Important:** If you use z/OS tools that intercept DCB abends (such as the B37 abend), ensure that you exclude the QMF spill file from such operations. Otherwise, QMF cannot properly manage the spill file, causing not only unpredictable results, but also difficulty in tracing and diagnosing any errors.

### **Increasing the storage group's volume space**

If the problem is caused by a lack of available space on the volumes of a control table storage group, add more volumes to this storage group with the Db2 ALTER STOGROUP statement.

### **Increasing the size of the CICS region**

If a QMF transaction runs out of virtual storage in the CICS region, the transaction might time out waiting for storage to become available. If this happens, ensure that the size of the CICS region is large enough to accommodate your users' needs. Make sure that you consider storage required for additional products you have installed.

### **Related concepts**

#### Solving resource contention problems

SELECT queries can time out when required rows are being updated by other operations. To increase concurrency of your applications and limit resource contention on Db2 for z/OS, you can set the QMF DSQEC\_CON\_ACC\_RES global variable according to your site's needs.

#### Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement

When Db2 dynamic statement caching is active, you can specify that Db2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This can result in more sharing and reuse of cached statements.

#### Addressing storage requirements

For the amount of storage required to copy the QMF Version 12.1 libraries from the distribution media using SMP/E, refer to the QMF program directory that is appropriate for your QMF version and national language.

### **Related tasks**

#### Capturing EXPLAIN information for dynamic statements

You can monitor the performance of eligible dynamic SQL queries by capturing EXPLAIN information for those queries. EXPLAIN data contains information about the access paths that are used to process SQL statements.

#### Enabling QMF queries to be eligible for query acceleration

Using a query accelerator that is available to the Db2 for z/OS database can significantly improve the performance of certain kinds of queries. QMF queries can be eligible to run on a query accelerator if they meet the prerequisites and conditions for query acceleration that are listed in the Db2 documentation.

#### Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting

Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

#### Spilling report data to extended virtual storage (TSO only)

In QMF for TSO, use extended storage for spill data unless the system on which QMF is running has very limited extended storage available.

## **Solving resource contention problems**

SELECT queries can time out when required rows are being updated by other operations. To increase concurrency of your applications and limit resource contention on Db2 for z/OS, you can set the QMF DSQEC\_CON\_ACC\_RES global variable according to your site's needs.

The values for this global variable support concurrent access resolution available in Db2 for z/OS Version 10 as well as the Db2 for z/OS Version 9 SKIP LOCKED DATA option.

### **Related concepts**

#### Solving storage problems

Users might notice slow performance in running queries or formatting reports when not enough virtual storage is available to retrieve all rows required by the operation. You can recover from this problem by increasing storage in one or more areas.

#### Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement

When Db2 dynamic statement caching is active, you can specify that Db2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This can result in more sharing and reuse of cached statements.

### **Related tasks**

#### Capturing EXPLAIN information for dynamic statements

You can monitor the performance of eligible dynamic SQL queries by capturing EXPLAIN information for those queries. EXPLAIN data contains information about the access paths that are used to process SQL statements.

#### Enabling QMF queries to be eligible for query acceleration

Using a query accelerator that is available to the Db2 for z/OS database can significantly improve the performance of certain kinds of queries. QMF queries can be eligible to run on a query accelerator if they meet the prerequisites and conditions for query acceleration that are listed in the Db2 documentation.

#### Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting

Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

## **Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting**

Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

### **About this task**

By default, QMF uses 4K buffers to maintain data rows for both single row and multirow fetch operations. For multirow fetch operations, if a data row length is greater than 2056 bytes, only one row will fit in a standard buffer per fetch. As a result, multirow fetch operations for long data rows are no faster than a single row fetch.

To change this behavior, administrators can increase the value of the DSQEC\_BUFFER\_SIZE global variable. Increasing the buffer size allows QMF to retrieve more data rows per fetch and can significantly improve performance by reducing database traffic.

## Related concepts

### Solving storage problems

Users might notice slow performance in running queries or formatting reports when not enough virtual storage is available to retrieve all rows required by the operation. You can recover from this problem by increasing storage in one or more areas.

### Solving resource contention problems

SELECT queries can time out when required rows are being updated by other operations. To increase concurrency of your applications and limit resource contention on Db2 for z/OS, you can set the QMF DSQEC\_CON\_ACC\_RES global variable according to your site's needs.

### Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement

When Db2 dynamic statement caching is active, you can specify that Db2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This can result in more sharing and reuse of cached statements.

### Enabling support for multirow fetch and insert

The DSQSMRFI parameter controls whether the database uses multirow or single-row fetch and insert.

## Related tasks

### Capturing EXPLAIN information for dynamic statements

You can monitor the performance of eligible dynamic SQL queries by capturing EXPLAIN information for those queries. EXPLAIN data contains information about the access paths that are used to process SQL statements.

### Enabling QMF queries to be eligible for query acceleration

Using a query accelerator that is available to the Db2 for z/OS database can significantly improve the performance of certain kinds of queries. QMF queries can be eligible to run on a query accelerator if they meet the prerequisites and conditions for query acceleration that are listed in the Db2 documentation.

Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

## Improving eligibility for the Db2 dynamic statement cache with literal concentration enablement

When Db2 dynamic statement caching is active, you can specify that Db2 exclude the literal constants that are in dynamic SQL statements when searching for a statement text match in the dynamic statement cache. This can result in more sharing and reuse of cached statements.

This Db2 feature can be achieved through QMF with the activation of the QMF global variable DSQEC\_CON\_CSWL. When DSQEC\_CON\_CSWL set to a value of '1', QMF will append the Db2 prepare attribute CONCENTRATE STATEMENTS WITH LITERALS to all user driven SQL SELECT statements. If activated, QMF commands such as RUN QUERY, DISPLAY TABLE, EXPORT TABLE and PRINT TABLE would be eligible for this Db2 performance improvement.

## Related concepts

### Solving storage problems

Users might notice slow performance in running queries or formatting reports when not enough virtual storage is available to retrieve all rows required by the operation. You can recover from this problem by increasing storage in one or more areas.

### Solving resource contention problems

SELECT queries can time out when required rows are being updated by other operations. To increase concurrency of your applications and limit resource contention on Db2 for z/OS, you can set the QMF DSQEC\_CON\_ACC\_RES global variable according to your site's needs.

## Related tasks

### Capturing EXPLAIN information for dynamic statements

You can monitor the performance of eligible dynamic SQL queries by capturing EXPLAIN information for those queries. EXPLAIN data contains information about the access paths that are used to process SQL statements.

### Enabling QMF queries to be eligible for query acceleration



Using a query accelerator that is available to the Db2 for z/OS database can significantly improve the performance of certain kinds of queries. QMF queries can be eligible to run on a query accelerator if they meet the prerequisites and conditions for query acceleration that are listed in the Db2 documentation.

**Improving QMF performance with the DSQEC\_BUFFER\_SIZE global variable setting**

Improve QMF performance by increasing the value of the DSQEC\_BUFFER\_SIZE global variable.

## Using diagnosis aids

Try to diagnose the problem by using a variety of diagnostic aids.

**Related concepts**

[Correcting common problems](#)

Check the list of common problems and possible solutions before you try a more in-depth diagnosis.

## Diagnosing your problem using message support

QMF issues various types of messages during a user's session, indicating either that QMF successfully completed the user's request or that an error occurred.

All QMF messages have a message number of the form DSQnnnnn, where nnnnn is a five-digit number.

To obtain the message number and more information about the error, press the Help key to display a message help panel. Each help panel has a panel number associated with it. If you report the problem to IBM, your IBM Software Support representative might need this number. To make sure that the number displays, set the global variable DSQDC\_SHOW\_PANID to 1: SET GLOBAL (DSQDC\_SHOW\_PANID=1

**Related reference**

[Db2 QMF Messages and Codes](#) Search for possible solutions for problems by message number or code.

**Determining which QMF function issued an error message**

You can use the QMF message number to determine which QMF component issued the message. By knowing the component, you can isolate the problem to a specific function.

The QMF functions and their associated ranges of message numbers are shown in the following table. The trace IDs are the same IDs that you use to trace QMF activity for each function.

In addition to the message numbers in this table, the following ranges of message numbers might be generated during QMF initialization:

DSQI0001 - DSQI0100  
DSQ90000 - DSQ99999

| Function                              | Trace ID | Message numbers                          |
|---------------------------------------|----------|------------------------------------------|
| Database services                     | I        | DSQ10000 - DSQ19999, DSQ30000 - DSQ39999 |
| Dialog command processing             | D        | DSQ20000 - DSQ29999                      |
| Display services                      | E        | DSQ40000 - DSQ49999                      |
| Common services and systems interface | C        | DSQ50000 - DSQ59999                      |
| Report formatting                     | F        | DSQ60000 - DSQ69999                      |
| Charting                              | P        | DSQ70000 - DSQ79999                      |
| Full-screen windows                   | G        | DSQ80000 - DSQ89999                      |

## Handling system error messages

A system error might indicate a system problem, a resource problem, or an unexpected condition.

These problems might be within QMF, the database manager, or some other software component. System errors are indicated by the following message:

Sorry, a system error occurred. Your command may not have been executed.

Press the Help key to display more information about the message.

All uncommitted changes to the database are rolled back when a system problem stops QMF. Error information about the system problem is written to the trace data, which is the only source of information for a system problem that stops QMF. The Q.ERROR\_LOG table contains information about a system error only if the error occurred while the database was still running.

## Related reference

[Db2 QMF Messages and Codes](#) Search for possible solutions for problems by message number or code.

## Handling SQL return codes

In some cases, the message that QMF displays might be associated with an SQL return code.

For example, suppose a user receives QMF message DSQ10422. This message is associated with SQL return code -30060, which has the following text:

```
RDB AUTHORIZATION FAILURE
```

The online help for all QMF messages that are associated with SQL return codes includes the contents of the SQLCA (SQL Communicates Area). To see the SQLCA output, press the Help key while the message is displayed in the QMF message area, then scroll to the end of the help text. For additional help with SQL return codes, see the Db2 documentation.

SQL queries that run multiple SQL statements can result in multiple SQL return codes (positive or negative). In these cases, the QMF message (and any associated SQL code) displayed when the query completes is associated with the last statement in the query only. You can view the QMF trace output to see all the SQL return codes and SQLCA information associated with all statements in the query.

You can use the DSQDC\_POS\_SQLCODE global variable to control how QMF handles positive SQL return codes. The following values are valid for this variable:

**2**

Allows users to display online help associated with the positive SQL code.

**1**

Logs the message associated with the SQL code to DSQDEBUG.

**0**

The message associated with the SQL code is not logged and no help text is provided.

## Related tasks

[The trace facility](#)

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

## Creating an interrupt to capture diagnostic information

You can use the QMF interrupt facility to gather information about a problem.

### About this task

By using the interrupt facility, you can produce an abend dump or cause trace information to be displayed or written to the DSQDEBUG data set.

In TSO, the QMF interrupt handler can be activated even when a QMF command is not running.

### Procedure

To interrupt QMF:



1. Log on to QMF under the user ID of the user whose problem you are diagnosing.
2. Re-create the problem.
3. At the point at which you want to start gathering information about the problem, create the interrupt.

For most system configurations, you can create an attention interrupt by pressing either the Attn key or a combination of the Reset and PA1 keys (or equivalent key combinations in a terminal emulation session). If these combinations do not work for you, see the appropriate information for your current system configuration to obtain more information about creating the interrupt.

The interrupt facility responds by displaying the following message:

```
DSQ50546 QMF command interrupted!   Clear screen and press enter.
```

*Figure 117. QMF interrupt handler prompt 1*

4. After the interrupt message appears, press the Clear and Enter keys (or an equivalent key combination in a terminal emulation session), as the message instructs you to do.

The following message appears:

```
DSQ50547 QMF command interrupted!   Do one of the following:
==> To continue QMF command,       type 'CONT'.
==> To cancel QMF command,         type 'CANCEL'.
==> To enter QMF debug,             type 'DEBUG'.
```

*Figure 118. QMF interrupt handler prompt 2*

5. Take one of the following actions:

- Enter CONT to return control to wherever you were before you caused the interrupt, as if the interrupt never occurred.
- Enter CANCEL to stop any command that is running at the time of the interrupt. The keyboard is unlocked, and QMF awaits your next command. However, it is not always possible to cancel a command.
- Enter DEBUG to capture diagnostic information. The following sample output is in response to entering DEBUG:

```
-- OK, QMF debug entered. QMF CSECT trace is:
   DSQDSUPV -> DSQDSUPX -> DSQEADAP -> DSQEMAIN -> DSQEINPT -> ENDTRACE
==> To continue QMF command,       type 'CONT'
==> To cancel QMF command,         type 'CANCEL'
==> To abnormally terminate QMF,   type 'ABEND'
==> To set QMF trace,              type 'TRACEALL' or 'TRACENONE'
```

*Figure 119. Diagnostic information captured by entering DEBUG in response to the interrupt prompt*

The trace information in the second line of this example indicates that, at the time of the interrupt, control was in CSECT DSQEINPT. Control reached this CSECT by passing successively through the CSECTs DSQDSUPV, DSQDSUPX, DSQEADAP, and DSQEMAIN.

6. Enter the tracing options that you want. Respond to the interrupt prompt panel by entering one of the following options:
  - Enter CONT to return control to wherever you were before you caused the interrupt, as if the interrupt never occurred.
  - Enter CANCEL to stop any command that is running at the time of interrupt. The keyboard is unlocked, and QMF awaits your next command. However, it is not always possible to cancel a command.
  - Enter ABEND to abnormally terminate QMF and produce an abend dump (if a DSQDUMP data set was allocated for the session).
  - Enter TRACEALL to cause QMF to start adding the most detailed level of tracing output to the DSQDEBUG data set. Control returns to wherever it was at the time of interrupt.
  - Enter TRACENONE to cause QMF to stop adding any trace output to the DSQDEBUG data set. Control returns to wherever it was at the time of interrupt.

## The trace facility

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

### Allocating the trace data set

Make sure that the DSQDUMP data set or DSQDEBUG data set is allocated, depending on how your run QMF.

Certain procedures in this information rely on abend information as well as trace information that QMF records in the DSQDEBUG data set.

- Trace information in TSO, ISPF, or native z/OS

Trace information is recorded in the DSQDEBUG data set. You can find abend dump information in the DSQDUMP data set. Make sure that this data set is allocated before you begin the QMF session. The data sets are allocated in the sample TSO logon procedure that is provided with QMF.

If these data sets are not allocated automatically before a QMF session, issue the following TSO statements before you invoke QMF for your diagnostic session.

```
ATTR DEBUG RECFM( F B A) LRECL(121)
ATTR DUMP RECFM( F B A) LRECL(125)
ALLOC DDNAME(DSQDEBUG) SYSOUT(A) USING(DEBUG)
ALLOC DDNAME(DSQDUMP) SYSOUT(A) USING(DUMP)
ALLOC DDNAME(SYSUDUMP) SYSOUT(A)
```

Figure 120. Allocating the data sets for TSO

If you are starting TSO as a Db2 for z/OS stored procedure, data sets for the trace output and other functions must be allocated in the JCL that starts the WLM address space from which QMF runs.

- Trace information in CICS

The trace is recorded in the DSQDEBUG data set. Allocate this data set in the CICS startup JCL. The trace can be shared between all users in the same CICS address space.

The trace data set is described as an extrapartition data set. It was defined when CICS installation job QMF1210.SDSQSAPE(DSQ1ECSO) was run. The following figure shows the definition information.

```
DEFINE TDQUEUE(DSQD)
DESCRIPTION(QMF Trace Info) GROUP(QMF1210E)
TYPE(EXTRA) BLOCKSIZE(6050) DDNAME(DSQDEBUG)
RECORDFORMAT(VARIABLE) TYPEFILE(OUTPUT) RECORDSIZE(121)
BLOCKFORMAT(BLOCKED)
```

Figure 121. Description (in a CICS table) of the trace data set

QMF trace data from all the QMF users in a single CICS region is written to a single trace data set. Each trace entry contains the terminal ID of the user who recorded it.

To look at the trace data set while the CICS region is active, you must close the trace data set that is using the CICS queue ID that you specified on the DSQSDBQN parameter (DSQD is the default). You can do close the trace data set by using the transaction CEMT supplied with CICS. For example:

```
CEMT I TDQUEUE(DSQD)
```

When the trace data set is closed, you can print or browse it from ISPF on TSO. When the trace data set is closed, no other records can be written by CICS users. QMF continues to operate in this state without recording trace records. To make the QMF trace available again, you can use the same CEMT transaction that you used to close the trace data set to open it again.

## Related tasks

### Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## Starting the trace facility

You start a QMF trace by having the user enter the SET PROFILE command with a tracing option or by updating the TRACE field in the user's profile. When you start QMF for TSO as a Db2 for z/OS stored procedure, tracing options are passed on the CALL statement that starts the interface.

## Before you begin

Be sure that a data set with a ddname of DSQDEBUG is allocated, as explained in [“Allocating the trace data set”](#) on page 356. The trace facility writes trace results into the DSQDEBUG data set, which can be printed or displayed. This data set is used for trace purposes only.

## Procedure

1. Decide on your tracing options.

With these options, you control what is traced and the level of detail.

When you specify a value of ALL on the DSQSDEBUG program parameter, all QMF activity is traced at the highest level of detail, including program failures that might occur during QMF initialization.

If the output exceeds the maximum size of a temporary storage queue, use a transient data queue.

2. Specify tracing options. During a QMF session, some set of tracing options is always in effect unless you specify a value of NONE on the DSQSDEBUG parameter. You can override current trace options in the following ways:

- Instruct the user to enter the following QMF command: SET PROFILE (T=*value*)

In the preceding command, *value* is ALL or a string that indicates QMF functions and their levels of detail in the trace output.

- Use SQL UPDATE statements for the TRACE field in the user's profile, which has the same effect as the previous method. Instruct the user to reconnect to the database to initialize the new values. For example, user JONES with password MYPW can enter: CONNECT JONES (PA=MYPW)

Users who do not have Db2 CONNECT authority can end the current QMF session and begin another to initialize the values.

## Related concepts

### Getting the right level of detail in your trace output

You can trace all QMF functions in detail or trace individual QMF functions.

### Viewing QMF trace data

DSQDEBUG holds the information that is recorded by the trace facility. It must be allocated before you start QMF if you want to use tracing.

### Starting QMF as a Db2 for z/OS stored procedure

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

## Related tasks

### Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### Getting the right level of detail in your trace output

You can trace all QMF functions in detail or trace individual QMF functions.

To trace all QMF functions at the most detailed level, use a value of ALL in the TRACE column of the Q.PROFILES table. If the results of this option are too large, you can limit the amount of trace output to a specific size by setting the DSQEC\_TRACE\_LIMIT global variable.

To trace individual QMF functions, use a different character string in the TRACE column of the Q.PROFILES table. Use letters for the QMF functions you want to trace and a number for the level of trace detail for each function. You must pair each letter with a number as follows:

- The value 1 traces a function at a medium level of detail.
- The value 2 traces a function at the highest level of detail.

When you use function-level tracing, only the functions you specify in the character string are traced. The following table shows the letter to use for each QMF function.

| Trace ID | QMF function that is traced                                                                                                   |
|----------|-------------------------------------------------------------------------------------------------------------------------------|
| A        | Application support services                                                                                                  |
| C        | Common services and systems interface                                                                                         |
| D        | Dialog command processing                                                                                                     |
| E        | Display services for parts of QMF such as Prompted Query, QBE, Table Editor, global variable lists, and database object lists |
| F        | Report formatting                                                                                                             |
| G        | QBE, Prompted Query, and Table Editor full-screen windows                                                                     |
| I        | Database services                                                                                                             |
| L        | Message and command logging                                                                                                   |
| P        | Charting (Interactive Chart Utility)                                                                                          |
| R        | Storage management functions                                                                                                  |
| U        | User exits, such as user edit exit routines or a governor exit routine                                                        |

For example, to trace message and command logging at the most detailed level, application support services at a medium level, and common services and systems interfaces at the most detailed level, use this command:

```
SET PROFILE (T=L2A1C2
```

The value of the TRACE option in the QMF profile is ignored when you start QMF as a Db2 for z/OS stored procedure. However, you can include a SET PROFILE (TRACE command in the initial procedure specified by the *object-name* parameter to change the level of trace detail for the duration of the stored procedure session as long as the *L2-destination* parameter is set to DSQDEBUG. When you issue a SET PROFILE command to change the trace settings, ensure that L2 is among the options specified on the SET PROFILE (TRACE command in the initial procedure if you want to continue tracing messages and commands.

Use the L1 and L2 trace records to precisely record user activities during a QMF session. A value of L1 writes records for all messages that are issued by QMF. L2 writes all the L1 records plus more records that describe the execution of QMF commands. Use the L2 trace code to log each command that a user

issued and how QMF responded to that command. The following example of a RUN QUERY command fail because the user names columns that are not in the table.

```
-----  
***** 93/12/15 20:39 *****  
-----  
USERID: KRIS  
AUTHORIZATION-ID: KRIS  
COMMAND TEXT:  
RUN QUERY  
-----  
***** 93/12/15 20:39 *****  
-----  
USERID: KRIS  
AUTHORIZATION-ID: KRIS  
MESSAGE NUMBER: DSQ12405  
MESSAGE TEXT:  
Column name DATE is not in table STAFF.  
&01: DATE  
&02: STAFF  
&09: -205  
-----
```

Figure 122. Using the L2 trace code to trace a user's commands and messages

Within the DSQDEBUG data set, the messages appear chronologically. When commands are included, they also appear chronologically and are intermixed with the messages. A message is associated with the command that precedes it in the data set or file.

QMF messages have variables for parts of the message that change, such as a table or column name. You can use the trace data to help a user decipher a message that includes variables. For example, the message that is shown in the previous example appears in *Db2 QMF Messages and Codes* as: Column &V1 is not in table &V2. The bottom half of the example shows that the value for &V1 in the message is DATE and that the value for &V2 is STAFF. Substitute these values into the message to help a user solve the problem.

These variables might also appear in the definition of the help panels that are associated with the error message. Use the variable values from the trace data together with the HELP command to reconstruct the message help panel if necessary.

### Related tasks

[Installing the QMF stored procedure interface \(TSO only\)](#)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

[Starting QMF for TSO from QMF for Workstation and running a procedure with logic](#)

In this example, you start QMF for TSO as a stored procedure from QMF for Workstation. You then run a procedure with logic that sets default values, retrieves global variables values, and runs a query.

### Tracing individual QMF modules

You can turn on a trace for certain modules using the SET PROFILE command and the module DSQUTRAC.

**Important:** Run a trace at the module level only under the guidance of IBM Software Support.

For example, you can trace the formatter buffer manager without tracing the line manager or the summary manager. The values for module-level tracing are:

- A value of 3 provides a detailed trace for specific programs in a component, and traces entry and exit for all other programs in the component.
- A value of 4 traces a module only.

To create a module-level trace, use the DSQUTRAC module to list the modules that you want to trace. Then assemble and link-edit the module. After the module is created, you must make it available. You can then run the following command: SET PROFILE (TRACE=F4

To trace only a specific module or set of modules, you must set the DSQEC\_TRACE\_MODULE global variable. In this variable, specify the name of the modules that you want to trace (you can specify up to six names separated by commas). Then, run the following command: SET PROFILE (TRACE=ALL. This will initiate tracing for only the modules specified in DSQEC\_TRACE\_MODULE. If you specify modules via the SET GLOBAL command from the command line, then module names must be enclosed in single quotes.

The value of the TRACE option in the QMF profile is ignored when you start QMF as a Db2 for z/OS stored procedure. However, you can include a SET PROFILE (TRACE command in the initial procedure specified by the *object-name* parameter to change the level of trace detail for the duration of the stored procedure session as long as the *L2-destination* parameter is set to DSQDEBUG. When you issue a SET PROFILE command to change the trace settings, ensure that L2 is among the options specified on the SET PROFILE (TRACE command in the initial procedure if you want to continue tracing messages and commands.

### Related tasks

#### Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

#### Starting QMF for TSO from QMF for Workstation and running a procedure with logic

In this example, you start QMF for TSO as a stored procedure from QMF for Workstation. You then run a procedure with logic that sets default values, retrieves global variables values, and runs a query.

### Viewing QMF trace data

DSQDEBUG holds the information that is recorded by the trace facility. It must be allocated before you start QMF if you want to use tracing.

If you are running QMF for TSO as a Db2 for z/OS stored procedure, you can allocate the data set to print or display it.

In CICS, depending on the number of users and the levels of detail at which their sessions are traced, the trace data might be long.

- Printing or displaying the trace data in TSO:

The DSQDEBUG data set can be allocated through the TSO logon procedure. However, if necessary, you can reallocate the file for printing by issuing statements like the following examples, which define DSQDEBUG as a PRINT file:

```
FREE FILE(DSQDEBUG)
ATTR DEBUG RECFM( F B A) LRECL(121)
ALLOC DDNAME(DSQDEBUG) SYSOUT(A) USING(DEBUG)
```

These statements allocate the trace file with fixed-length, 121-character records whose first byte is an ANSI carriage-control character. The trace information is formatted with 120 characters to the line, not including the ANSI control character.

**Important:** If you allocate output from DSQDEBUG to go to the HOLD queue, to release the output to the OUTPUT queue, you must issue the following TSO command: FREE DDNAME(DSQDEBUG)

You can also issue the following statements to allocate (or reallocate) the DSQDEBUG data set as a sequential data set that can be displayed by using an online editor. The following example statements allocate the trace file with fixed-length, 81-character records whose first byte is an ANSI carriage-control character. The trace information is formatted with 80 characters to a line, not including the ANSI control character.

```
FREE FILE(DSQDEBUG)
ATTR DEBUG RECFM( F B A) LRECL(81)
ALLOC DDNAME(DSQDEBUG) DSNAME(DEBUG.LIST) NEW KEEP
```

- Printing or displaying the trace data in CICS:

The trace is recorded in the DSQDEBUG data set. Allocate this data set in the CICS startup JCL.

If you have a warning or a system error during QMF initialization, you must look at the QMF trace data set to understand the reason for the error. QMF trace data from all the QMF users in a single CICS region is written to a single trace data set. Each trace entry contains the terminal ID of the user who recorded it.

### Related concepts

Allocating the trace data set

Make sure that the DSQUDUMP data set or DSQDEBUG data set is allocated, depending on how your run QMF.

### Related tasks

Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

### Determining the QMF service level

Running an SMP/E report against the target or distribution zones is always the best way to determine the service level. However, you can also determine the QMF service level in other ways.

### Procedure

You can determine the QMF service level in the following ways:

- Issue the SHOW command

The SERVICE option of the SHOW command returns the service information for the specified module in a message. For example, SHOW SERVICE .DSQCSID0 returns a message similar to the following message:

```
OK, DSQCSID0 V11R2.00 09/09/16 13:39 WIM11299
```

- Look at trace data to get the service level

The trace output can help you obtain the service level of your QMF installation. The service level information is automatically displayed when an abend occurs.

You can obtain the QMF service level in the following way when you experience non-abend problems:

- a) Set the trace for the highest level of detail by specifying a value of ALL for tracing.  
You can set this value on the DSQSDEBUG parameter when you invoke QMF or use the following command to set the value in the Q.PROFILES table: SET PROFILE (T=ALL
- b) Reset the trace to a value of NONE by using the following command: SET PROFILE (T=NONE
- c) Exit QMF.
- d) Examine the DSQDEBUG file.

The resulting trace shows the program with its version, date, and time. The trace can also show Authorized Program Analysis Report (APAR) numbers if the module has one or more Program Temporary Fixes (PTFs) applied, as in the following trace example:

```
** DSQFQWRM: ENTERED FROM DSQFMCTL ***  
V10R1.00 10/10/25 12:00 PNnnnnn
```

APAR PNnnnnn is the most recent APAR for which service was applied, where the nnnnn characters represent numbers.

If you see a blank value in place of PNnnnnn or you see WIMnnnnn, the module is either at "base level" or no service was applied or is available.

- Run a batch job to determine the service level

Another way to determine the QMF service level is to submit a batch job like the following one:

```
//ROBINK JOB (&SYSUID,090,T48,G249),ROBIN,  
// NOTIFY=ROBIN,TIME=2,MSGCLASS=H,USER=&SYSUID
```

```
//AMBSTEP EXEC PGM=AMBLIST
//SYSPRINT DD SYSOUT=*
//DDDD1 DD DSN=QMFDEV.QMF1210.SDSQLOAD,DISP=SHR
//SYSLIB DD DUMMY
//SYSIN DD *
LISTIDR DDN=DDDD1,MODLIB
```

This job produces a report that shows the latest PTF applied to each CSECT within a load module.

### What to do next

Make sure that the appropriate level of service has been applied to the DSQPNLx runtime panel file, where x is your language identifier. Occasionally, after PTFs have been applied, the equivalent service updates are overlooked and are not applied to the runtime panel file, which can cause unpredictable behavior.

To check the service level of the DSQPNLx runtime panel file, display the QMF home panel, press PF1, and select **About QMF**. The service level of DSQPNLx is provided.

If the service level of DSQPNLx is not current, contact your system administrator to determine if QMF1210.SDSQSAPE(DSQ1xNPL) was run after the latest PTFs were applied.

### Turning off the trace facility

After you capture diagnostic details using the trace facility, you might want to turn tracing off, because the storage queue for the trace data can fill up very quickly.

### About this task

If you leave tracing on until you end the QMF session, when you start QMF the next time, the tracing is set to NONE by default. The program parameter DSQSDEBUG controls this tracing when QMF is started.

### Procedure

- To turn tracing off, issue the following command from within QMF: SET PROFILE (T=NONE

### Related tasks

[Setting tracing options](#)

QMF provides a trace facility that helps track QMF and user activity.

## Examining error log reports

The Q.ERROR\_LOG table is a QMF control table that logs information about resource and other problems.

The structure of the table is shown here.

| <i>Table 80. Structure of the Q.ERROR_LOG table</i> |                      |                |                                                                                                                                 |
|-----------------------------------------------------|----------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------|
| Column name                                         | Data type and length | Nulls allowed? | Function/values                                                                                                                 |
| DATESTAMP                                           | CHAR(8)              | no             | The date on which the error occurred. It is in the form <i>yyyymmdd</i> .                                                       |
| TIMESTAMP                                           | CHAR(5)              | no             | The time at which the error occurred. It is in the form <i>hh:mm</i> , where <i>hh</i> is the hour and <i>mm</i> is the minute. |



Table 80. Structure of the Q.ERROR\_LOG table (continued)

| Column name | Data type and length                                                                                                                                  | Nulls allowed? | Function/values                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| USERID      | VARCHAR(128)<br><b>Exception:</b> When you are connected to a DB2 for VSE and VM database, the data type and length for the USERID column is CHAR(8). | no             | In QMF for TSO, this column contains the authorization ID or the TSO logon ID of the user who experienced the error.<br><br>In QMF for CICS, this is the CICS terminal ID of the user who experienced the error.<br><br>For errors experienced when you are running QMF for TSO as a Db2 for z/OS procedure, this column contains the authorization ID of the user who started the WLM-managed address space from which QMF runs. |
| MSG_NO      | CHAR(8)                                                                                                                                               | no             | The QMF message number that was issued with the error.                                                                                                                                                                                                                                                                                                                                                                            |
| MSGTEXT     | VARCHAR(254)                                                                                                                                          | no             | Text of the message. SQL errors might have data from the SQLCA in this column.                                                                                                                                                                                                                                                                                                                                                    |

A long error message might need more than one row of the table to represent it. If it does, the values of every column except the MSGTEXT column repeat. Within the MSGTEXT column, each row carries a fragment of the message. A fragment begins with 1), 2), 3), and so on, to indicate its relative position in the message.

To help diagnose problems, you can query the Q.ERROR\_LOG table for information about errors as shown in the following example.

```
SELECT TIMESTAMP, MSG_NO, MSGTEXT
FROM Q.ERROR_LOG
WHERE USERID = 'id'
      AND DATESTAMP = 'date'
      AND TIMESTAMP BETWEEN 'time1' AND 'time2'
ORDER BY TIMESTAMP, MSG_NO, MSGTEXT
```

Figure 123. Querying the error log for problem information

### Related concepts

#### Starting QMF as a Db2 for z/OS stored procedure

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

### Related tasks

#### Installing the QMF stored procedure interface (TSO only)

QMF Version 12.1 allows you to start QMF for TSO as a Db2 for z/OS stored procedure. Users do not need to log on to QMF for TSO to perform tasks.

## Using diagnostics native to the environment

You might need to diagnose abends using diagnostic facilities in TSO, z/OS, or CICS.

In CICS, abend information is recorded in the DFHDMPx data set. This data set should be allocated in the CICS startup JCL. Most QMF programs contain a stamp that you can use to help identify them in diagnostic output. Here is an example.

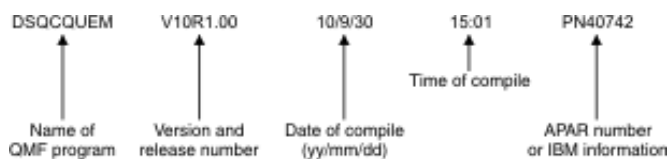


Figure 124. Example of a stamp that identifies a QMF program

---

### Diagnosing problems in TSO

To diagnose an abend in TSO, you might need to use procedures in the z/OS information center or you might be able to use the QMF abend handler.

When QMF starts, it establishes an abend handler. If QMF fails, the abend handler gets control, records the error, and cleans up the environment. After completion, the abend handler returns control to the operating system and allows it to continue with the abnormal termination process.

If an abend occurs while processing a user edit code or while executing the governor exit routine, additional areas appear in the dump to assist with problem diagnosis:

- For problems with user edit exit routines: DXEECS, the input area, and the result area are added to the output.
- For problems with the governor: DXEXCBA and DXEGOV are added to the output.

### Diagnosing problems in CICS

To diagnose an abend in QMF, you might need to use procedures in the CICS documentation.

Because another program might have caused QMF to abend, these procedures in the CICS documentation can help you find much of the information you need in a CICS dump of the transaction. A transaction dump shows detailed activity of the programs that were running in the CICS region at the time of the abend.

The program that caused the abend might be QMF or it might be another program. You can use the CICS Execution Diagnostic Facility (CEDF) to help you diagnose a QMF abend if the QMF diagnostic facilities explained in this information do not contain enough detail about the cause of the error.

- Identifying QMF in CICS diagnostic output

If you use CICS diagnostic facilities to help you diagnose an abend in QMF, the following information might help you identify QMF programs in CICS output.

- QMF program names begin with the prefix DSQ.
- QMF is an Assembler-language program and issues standard Assembler calls, not CICS LINK statements.
- QMF issues standard EXEC CICS statements for all system services when running in CICS.
- QMF uses an internal call interface to the GDDM product.
- QMF issues standard EXEC SQL statements to the database.
- QMF does not issue any EXEC CICS ABEND commands.

- Defining the display for a CICS abend message

In some cases, such as if QMF abends or when the operator cancels the transaction, CICS sends a message to the user's display indicating the abnormal ending. Because QMF is a full-screen application that uses GDDM to provide display services, you need to define to CICS how you want the abend message displayed.

Using the CICS Resource Definition Online (RDO) facility, set diagnostic display attributes of the CICS error message in the CICS TYPETERM definition. A TYPETERM is a partial terminal definition that makes it easy for you to define many terminal displays with one definition. The following example shows diagnostic display attributes you might use. This definition displays the message at the bottom of the screen, beneath the QMF message line. The message appears in red, underlined, and with a higher

intensity than the rest of the screen display. This definition is useful if you defined the QMF transaction to time out when the user does not enter input for a certain amount of time. In this type of transaction timeout, the QMF display remains on the screen, so the message is readable only at the bottom of the screen.

```

DIAGNOSTIC DISPLAY
ERR Last line      : Yes           No | Yes
ERRIntensify      : Yes           No | Yes
ERRColor          : Red          NO | Blue | Red | Pink | Green
                  | Turquoise | Yellow | Neutral
ERRHighlight      : Underline    No | Blink | Reverse | Underline

```

Figure 125. TYPETERM specification for CICS diagnostic display

## Reporting a problem to IBM

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Software Support. IBM Software Support provides assistance with product defects, answers FAQs, and helps users resolve problems with the product.

Before you report a problem to IBM, check the IBM Software Support website to see if the problem has already been reported. For unreported problems, IBM Software Support prepares an Authorized Program Analysis Report (APAR), which includes useful information about how to solve the problem.

### Searching previously reported problems

You can often find solutions to problems by searching IBM knowledge bases.

Search the IBM Software Support website by constructing a string of search words that describes your problem. Every string of search words for QMF begins with the component ID 566872101 and the release number that matches the QMF national language environment in which you experienced the problem.

*Table 81. Release numbers for QMF base product and NLFs*

| <b>NLF</b>           | <b>ID</b> |
|----------------------|-----------|
| Brazilian Portuguese | B5B       |
| Canadian French      | B5G       |
| Danish               | B55       |
| English              | B10       |
| French               | B56       |
| German               | B57       |
| Italian              | B58       |
| Japanese             | B59       |
| Korean               | B5A       |
| Spanish              | B5C       |
| Swedish              | B5D       |
| Swiss French         | B5E       |
| Swiss German         | B5F       |
| Uppercase English    | B51       |

## Working with IBM Software Support

If you are having trouble diagnosing the problem and have used all available diagnosis aids, contact IBM Software Support to report the problem.

To help diagnose the problem, your Software Support representative might need some information that provides more details about the problem. For example, if you call to report an abend in QMF, you might need to supply some information about CSECTs of the program that you suspect might have caused the error. In many cases, you can find this type of information using the trace facility.

### **Related tasks**

The trace facility

QMF provides a facility that traces QMF activity during a user's session. Trace output from the facility can help you analyze errors such as incorrect or missing output, performance problems, or loops.

---

## Appendix A. Summary of changes in prior releases

QMF Version 12 Release 1 supports migrations from Version 8.1 NFM, Version 9.1 NFM, Version 10.1, Version 11.1 and Version 11.2. Enhancements in prior releases might affect how you plan your QMF migration.

Choose the topic associated with the release from which you are migrating and read topics for every subsequent release.

### Related concepts

[Migration and fallback considerations](#)

After you install or migrate to QMF Version 12.1, you might need to access and use objects from earlier releases of QMF to thoroughly test the new installation under site-specific conditions.

---

## Version 11.2 changes

These are enhancements to QMF Version 11 Release 2.

### Enhanced QMF Editor

The QMF Enhanced Editor feature brings creating and editing QMF SQL queries and procedures to a whole new level. For SQL queries, the QMF Enhanced Editor provides customizable highlighting and formatting for SQL syntax, reserved words, functions, and data types. Parenthesis checking is also available.

The Query Assist feature provides table name suggestions, column name and data type information, and suggested column value information. A preview pane is available for immediate SQL results checking.

The QMF Enhanced Editor can be used for QMF linear procedures and procedures with logic. Highlighting similar to that found in the query object is also available.

The QMF Enhanced Editor is accessed from the QMF command line by entering the QMF EDIT command with the EDITOR keyword set to 'EE'. The new DSQEC\_EDITOR global variable provides a customizable override of the EDITOR keyword specifications.

### Get organized with QMF folders

QMF queries, procedures, forms, and analytics can now be organized into groups called folders.

The LIST, SAVE, and ERASE commands have been updated to operate with the FOLDER keyword, which allows you to view and manipulate QMF objects in a more defined way. By using the SAVE command, you can automatically create a new folder by saving your first object to it. The RENAME command has been updated to rename FOLDER objects and to rename FOLDER references when a QMF object is renamed.

The new DSQEC\_CURR\_FOLDER global variable allows you to define your own default value for the FOLDER keyword.

### QMF Analytics for TSO enhancements

Three new models, Wilcoxon Signed-Rank Test, Mann-Whitney U Test, and F-Test have been added to the statistics modeling capabilities of the QMF Analytics for TSO feature.

The ability to choose columns for use in analytical analyses has been improved with enhanced data type targeting and information. The columns that are shown are based on model requirements; you can see a sample of values such as maximum and minimum to aid in column input to the model.

User-defined mapping capability has been added to QMF Analytics for TSO. OpenGIS WKT map definitions are available in either Db2 tables or exported data sets, which can be read by QMF Analytics for TSO to format user-specific maps. Maps for Africa, North America, South America, and Germany have been added to the existing library of predefined maps.

Saving analytics has been updated to allow listing of existing analytics objects.

### **SQL enhancements**

Enhancements for SQL queries include the following changes:

- The following special registers can now be set from within a QMF SQL query:

#### **CURRENT APPLICATION COMPATIBILITY**

Specifies the compatibility level support for dynamic SQL.

#### **CURRENT GET\_ACCEL\_ARCHIVE**

Specifies whether a dynamic SQL query uses archived data if the query refers to a table that is archived on an accelerator server.

- By using the new DSQSFISO program parameter option, scalar date and time functions such as CHAR will now reflect the actual date and time format of the database and not the ISO format precompiler option of QMF, which was the format used in previous releases. The new DSQAO\_DATE\_FORMAT and DSQAO\_TIME\_FORMAT global variables inform the user of the default date and time formats used by the database. The new DSQAO\_DSQSFISO global variable contains the value of the DSQSFISO program parameter setting.

### **QMF command and command behavior**

Enhancements to QMF commands and command behavior include the following changes:

- The process of saving database tables, which was traditionally accomplished by using the QMF SAVE DATA command, has been improved for increased performance and reduced storage requirements. By using the new TABLE keyword on the RUN QUERY command, you can now SAVE DATA without having to return and complete a data object. The RUN QUERY with the TABLE keyword operates completely within the database to both retrieve data and insert rows without returning a report to the user.
- The SPACE keyword has been added to the SAVE DATA, IMPORT TABLE, and RUN QUERY commands (when the TABLE keyword is specified) to allow you to override the SPACE specifications that are usually set in your profile. Specific databases and tablespaces can be targeted for saving data without updating a user's profile.
- The ACCELERATOR keyword has been added to the SAVE DATA, IMPORT TABLE, and RUN QUERY commands (when the TABLE keyword is specified) to allow the creation of Db2 accelerator-only tables. The new DSQEC\_SAV\_ALLOWED global variable controls whether an accelerator, traditional tablespace, or both can be used. The new DSQEC\_SAV\_ACCELNM global variable has been added to provide a default for the ACCELERATOR keyword.
- A new command, RENAME, has been added to give you the ability to rename Db2 tables or QMF objects in a single step.
- The QMF EDIT command has been updated to load saved queries and procedures directly from the database rather than requiring that queries and procedures be already located in temporary storage.
- The new DSQCP\_RMV\_BLANKS global variable allows you to retain trailing blanks on VARCHAR and VARGRAPHIC columns while using the QMF EDIT TABLE command. Columns that contain relevant trailing blanks will be preserved after editing by the QMF Table Editor.
- Exporting QMF reports can now be done with no control information. You can use the EXPORT REPORT command with the DATAFORMAT keyword set to the new value 'TEXT' in order to get preallocated data sets that contain formatted QMF reports. The report output is suitable for text attachments or easy reading.

### **Default form overrides**

By using following new the global variables, you can override QMF default form edit codes:

- DSQDC\_EC\_CHAR
- DSQDC\_EC\_NUM
- DSQDC\_EC\_DEC

- DSQDC\_EC\_DATE
- DSQDC\_EC\_TIME

Default formatting can be controlled by global variable settings with no saved customized forms at all, which can reduce the amount of specialized forms in which only edit code changes exist.

### **Query by example (QBE) enhancements**

The QBE interface has been updated to support 128-character owner and table names and 30-byte column names. The DRAW, IMPORT, and EXPORT commands for QBE objects have also been updated to support increased owner, table, and column name lengths.

### **Application programming support**

Documentation about session-cancelling topics within the QMF governor interface has been improved.

### **QMF administration and serviceability**

This release of Db2 QMF includes the following serviceability and administration enhancements:

- The DSQEC\_SAV\_ALLOWED global variable allows administrators to control the implementation of the SAVE DATA, IMPORT TABLE, or RUN QUERY to TABLE commands. This global variable controls whether you can save data to traditional tablespaces, an accelerator, or block the use of the commands. The global variable allows you to specify defaults or to override defaults with keyword values.
- The DSQEC\_SPAC\_OVERRIDE global variable allows administrators to control whether the SPACE parameter can be used to override the Q.PROFILES.SPACE values on SAVE DATA, IMPORT TABLE, or RUN QUERY to TABLE commands.
- Storage sizes for QMF procedures and queries have been optimized so that a minimal amount of space is used for storage within the QMF catalog.
- Improved default tracing diagnostics are now produced for default trace situations. QMF release and database connections will be easily discovered in default tracing. QMF trace size controls have been added with the new DSQEC\_TRACE\_LIMIT global variable .
- The new 'SHOW SERVICE' command displays service-level information.
- DSQPNLE panel service-level information can be quickly determined from the 'About QMF' panel, which is accessed from the main help system.

For additional information about any of these enhancements, see the QMF Version 11 Release 2 information in IBM Knowledge Center at: <http://ibm.com/support/knowledgecenter/SS9UMF/welcome.html>.

### **Removal of DSQ0EQ24**

QMF sample query SAMPLE\_SYNONYM, which existed in QMF.SDSQSAPE member DSQ0EQ24, has been removed from the product.

## **Version 11.1 changes**

---

These are enhancements to QMF Version 11 Release 1.

### **QMF Analytics for TSO**

QMF Analytics for TSO allows you to analyze query results that are returned by QMF for TSO. QMF Analytics for TSO can be started only from within QMF for TSO, and it provides statistical analysis, forecasting functions, and additional chart types, all from an easy-to-use, menu-driven interface.

In QMF Analytics for TSO, you can save a chart or statistical analysis specification as an ANALYTIC object by pressing the Save function key. You can then use the saved specification in a RUN QUERY command by

including the ANALYTICS option in the command. You can also list, display, and erase an ANALYTIC object through the LIST, DISPLAY, and ERASE commands.

### Improved report storage management

Enhancements to report storage management include the following changes:

- New maximum for storage allocation

The maximum storage that can be specified on the DSQSBSTG parameter has been increased to 2GB. You can also now specify a percentage instead of a fixed value for the DBSQSBTG parameter. This percentage specifies the maximum percentage of available storage that is to be used for reports.

- Dynamic handling of storage

Prior releases of QMF assessed storage requirements for reports only once during a session, after the first report was returned. In this release, QMF assesses and manages storage requirements dynamically throughout a session when a new report is requested.

- New global variables for determining storage settings

Four new global variables contain the values of storage management parameters that were used for QMF initialization: DSQAO\_DSQSBSTG, DSQAO\_DSQSRSTG, DSQAO\_DSQSPILL, DSQAO\_DSQSPTYP. The information that is stored by these global variables can help administrators with storage management and troubleshooting.

### Enhancements for large data types

Enhancements for large data types include the following changes:

- LOB data enhancements

- QMF now returns up to 2 GB of CLOB and BLOB data and up to 1 GB of DBCLOB data for DISPLAY commands and SELECT queries. To display all of the data with these data types up to their maximums, set the column width on the QMF form to the maximum width. For CLOB and BLOB data, the maximum width is 32767, and for DBCLOB data, the maximum width is 16383. Then, use the wrapping edit code (CW, BW, or GW) that is appropriate for the data type.
- You can now import, export, and save LOB data through the QMF IMPORT, EXPORT, and SAVE DATA commands.
- Preferences for saving and retrieving LOB data can now be controlled by the following new global variables:
  - DSQEC\_LOB\_RETRV can be used to control how LOB data is retrieved and displayed by the following QMF commands: DISPLAY TABLE, EXPORT TABLE, EXPORT DATA, SAVE DATA, PRINT TABLE, PRINT REPORT, and RUN QUERY.
  - DSQEC\_LOB\_SAVE can be used to control whether users can save LOB data through the SAVE DATA or IMPORT TABLE commands.
  - DSQEC\_LOB\_COLMAX can be used to specify the maximum size of LOB data to retrieve, in kilobytes (up to 2 GB). Global variable DSQEC\_LOB\_RETRV must be set to either 1 or 3 for DSQEC\_LOB\_COLMAX to be effective. If a user tries to work with data containing a LOB column that is larger than the maximum specified by DSQEC\_LOB\_COLMAX, an error is issued and no report data is displayed.
- QMF now supports the use of three-part table and view names to access LOB data at remote locations in a distributed unit of work environment. To enable the use of three-part names, you must set the DSQEC\_LOB\_RETRV global variable to 2 or 3.

- XML data enhancements

- In prior versions of QMF, entire XML documents were read into buffer storage when the IMPORT command was issued for data in XML data format. QMF Version 11 allows one row at a time to be read into buffer storage. This enhancement results in reduced storage usage and can improve processing of data in XML data format depending on the size of the imported data.



- QMF now supports DISPLAY, EXPORT, IMPORT, SAVE, SELECT, and RUN operations with XML data when connected to Db2 for Linux, UNIX, and Windows Version 9.5 and later or DB2 for iSeries Version 7 and later.
- Extended storage support for fetching LOB and XML data

In prior releases, below-the-bar storage was used for fetching LOB and XML data. Because this release increases the maximum size that QMF supports for these data types, large amounts of storage might be used below the bar. Instead, LOB and XML data is now fetched and then stored in storage above the bar until it is needed. At that point, the data is moved to below-the-bar storage and deleted from below-the-bar storage when it is no longer needed.

### **Broader support for temporal data**

QMF Version 10 included support for selecting, creating, and updating temporal data through SQL queries. Temporal data is now additionally supported by the following commands and objects:

#### **Prompted queries**

The Specify panel of Prompted Query now contains a Time Period option that you can use to add a time period specification to a query for a temporal table or view.

#### **LIST command**

The LIST command now lists both temporal and history tables. Using DESCRIBE on a Table List panel displays information about the table, including the table's subtype. "History Table" is now a valid subtype. On Db2 for Linux, UNIX, and Windows, history tables display with a "Table" subtype.

#### **EXPORT command**

Although temporal tables cannot be exported directly from the database through the EXPORT TABLE command, you can export temporal data from QMF temporary storage through the EXPORT DATA command. Temporal data that is exported has no association with history data.

#### **IMPORT and SAVE commands**

You can use the IMPORT and SAVE commands to append or replace existing temporal data by specifying ACTION=APPEND or ACTION=REPLACE. However, you cannot use these commands to create a new temporal table. IMPORT TABLE *newtable* and SAVE DATA AS *newtable* create new non-temporal tables.

### **SQL query enhancements**

Enhancements for SQL queries include the following changes:

- To control the unit of work, you can now include COMMIT statements in queries that contain multiple SQL statements. If the user's QMF profile has the CONFIRM option set to YES to enable confirmation prompting, a prompt is displayed for each COMMIT statement in the query.
- SQL queries that contain multiple statements can now include one SELECT statement. In prior releases, an SQL query that contained a SELECT statement could not contain any other statements.
- If confirmation prompts are enabled, prompts are displayed for SELECT statements that modify the database through an embedded INSERT, DELETE, MERGE, or UPDATE statement. Confirmation prompts for this type of database change are supported on all databases that support this type of SELECT statement. Confirmation prompts were previously not displayed for any SELECT statements.
- This release of QMF provides additional support for Db2 special registers.
  - A new global variable, DSQEC\_EXPL\_MODE, can be used to set the CURRENT EXPLAIN MODE special register. This special register controls EXPLAIN behavior for dynamic SQL statements that are run through the RUN QUERY command.
  - The following special registers can be set from within a QMF SQL query:

#### **CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION**

Controls which materialized query tables that are enabled for optimization are considered during the optimization process of dynamic SQL queries.

### **CURRENT QUERY ACCELERATION**

Specifies whether queries are to be considered for query acceleration by the Db2 optimizer.

### **CURRENT TEMPORAL BUSINESS\_TIME**

Specifies a `TIMESTAMP(12)` value that is used in the default `BUSINESS_TIME` period specification for references to application-period temporal tables.

### **CURRENT TEMPORAL SYSTEM\_TIME**

Specifies a `TIMESTAMP(12)` value that is used in the default `SYSTEM_TIME` period specification for references to system-period temporal tables.

- A new global variable, `DSQAO_DSQSMRFI`, can be used to check the value of the `DSQSMRFI` program parameter.
- QMF now handles the nullable `ROWID SQLTYPE 905`, which Db2 returns under some circumstances when a `LEFT OUTER JOIN` is included in a query.
- QMF users can now call a stored procedure that references a `DISTINCT` data type.
- This release of QMF supports the setting of Db2 global variables through a `SET` statement if the new `DSQEC_KEEP_THREAD` global variable is set to 1. Db2 global variables are named memory variables that you can access and modify through SQL statements. A Db2 global variable is associated with a specific application context and contains a value that is unique to that application scope. The scope of global variables is similar to that of Db2 special registers because after they are created, the definitions of global variables are shared across different Db2 connections. However, each connection maintains its own instance of each global variable, so the variable content is only shared among SQL statements within the same connection.
- `ROWLIMIT` processing is now done by Db2 instead of QMF. This change provides a performance improvement in QMF when users issue `RUN QUERY` with `ROWLIMIT` for complex queries. In particular, this enhancement can provide performance improvements for large result sets that have an `ORDER BY` clause in the query.

### **QMF commands and command behavior**

Enhancements to QMF commands and command behavior include the following changes:

- Enhancements to `EXPORT` operations
  - This release adds the ability to export data in CSV (comma-separated value) format. You can then download the exported data to your workstation where data in CSV format can be used with applications such as Microsoft Excel.
  - Prior releases required SQL query and procedure objects to be exported to data sets with an `LRECL` of 79 bytes. These objects can now be exported to data sets with an `LRECL` of 79 to 32,760 bytes for new and existing data sets, allowing use of existing `DFSMS DATA` classes. A new global variable, `DSQEC_DSLRECL1`, can be used to set the record size to use when exporting an SQL query or QMF procedure to a new data set.
  - In QMF, you can set the number of primary and secondary tracks for data sets that are used to store the results of the `EXPORT` command. In prior releases, you could specify up to 65,535 tracks for PS and PDS data sets. This limit is now increased to 16,777,215 tracks. The setting of QMF global variable `DSQEC_DSALLOC_PRI` controls this allocation. This support also allows QMF to function with extended address volumes (EAV).
- Increased control for setting an object's `LAST_USED` timestamp in the `Q.OBJECT_DIRECTORY` table of the QMF catalog

The `DSQEC_LAST_RUN` global variable now includes another option that restricts the updates of the timestamp to `RUN` commands only. To support this change, the `DSQUOPTS` initialization routine was also updated.

- Improvement to the `DPRE` command synonym

`DPRE` is a command synonym that provides a print preview in QMF for TSO. In previous releases, the synonym ran a QMF procedure, thus replacing any procedure object that was in temporary storage

when the synonym was issued. The synonym definition and associated code now runs as a REXX exec, which allows the current procedure object to remain in temporary storage.

- New SHOW ANALYTICS command

This command provides access to the new QMF Analytics for TSO product, which is described in more detail in [“QMF Analytics for TSO”](#) on page 369.

### **Application programming support**

In prior releases, QMF set the z/OS program mask to get control on operations that could cause fixed-point or decimal overflow. Because problems can occur in other applications when QMF leaves the program mask set, the current mask is now saved and set to zero before calls to the following services or commands:

- QMF governor exit routines
- REXX interfaces for forms or procedures with logic that require REXX
- EDIT PROC and EDIT QUERY commands
- The TSO command and commands that require ISPF

Also, when programs call QMF, QMF now preserves the caller's program mask and initializes the program mask for use by QMF. When QMF returns control to the caller, the caller's program mask is restored.

### **Enhancement for callable interface applications**

A new TRACE command can be used to add trace information in applications that use the callable interface except for applications that are written in REXX.

### **Global variable management enhancements**

Enhancements to QMF global variable management include the following changes:

- Control of initial global variable settings

Administrators can populate a new Q.GLOBAL\_VARS table with initial values for global variables and specify whether those values can be updated by users. This table is installed as an empty table during QMF installation and is read at QMF initialization.

- Persistence of global variable values across sessions

A new global variable, DSQEC\_USERGLV\_SAV, controls whether values of user global variables and global variables that start with “DSQ” are saved from session to session. Values that are to be saved are stored in the Q.GLOBAL\_VARS table and associated with user IDs. See QMF Reference for more information about this global variable.

- Saving user input

A new global variable, DSQEC\_SESSGLV\_SAV, controls whether user input in some data entry fields on some panels is saved within and across QMF sessions. See QMF Reference for more information about this global variable. User input is saved through new global variables that are named with a DXY prefix.

### **Support for EAV DASD**

z/OS Version 1 Release 11 introduced extended address volumes (EAV) support for extended format sequential data sets and VSAM data sets. QMF Version 11 was enhanced to support these volumes.

### **Thread management**

A new QMF global variable, DSQEC\_KEEP\_THREAD, provides the ability to retain a Db2 thread that is used by QMF until the end of the QMF session. In previous releases of QMF, each Db2 thread that was used by QMF was deleted at the end of each query.

## Version 10.1 changes

---

These are enhancements to QMF Version 10.1.

### Stored procedure interface to QMF for TSO

A new interface allows you to start QMF for TSO as a Db2 for z/OS stored procedure. With this new feature, any software program that can call a Db2 for z/OS stored procedure can now start QMF for TSO and receive report output back in up to 20 result sets.

### Query enhancements

Query enhancements include:

- Support for multiple SQL statements in a single SQL query

Each SQL query can now contain multiple statements that involve database maintenance, such as UPDATE, INSERT, ALTER, CREATE, DROP, EXPLAIN, and others. Statements that return results, such as SELECT or CALL, remain limited to one per SQL query, as does the CREATE PROCEDURE statement.

A new global variable, DSQEC\_RUN\_MQ, allows you to set whether multiple SQL statements will be allowed. When multiple statements are allowed, all of the statements in the query are issued as a single unit of work. If one statement in the query fails, database changes made by statements prior to the failing statement are rolled back and subsequent statements are not run. Some statements, such as SET, apply to aspects of the QMF session and therefore are not rolled back in the case of a failure.

Substitution variable values and responses to confirmation prompts apply to all SQL statements in the query.

- Support for running QMF SQL queries over 32 KB long

The maximum size of an SQL query that can be run by the RUN QUERY command has been increased to 2 MB for SQL queries directed to Db2 for z/OS and 65 KB for SQL queries directed to DB2 for iSeries or Db2 for Linux, UNIX, and Windows. You can activate this feature by setting the DSQEC\_SQLQRYSZ\_2M global variable. SQL queries directed to DB2 for VSE and VM remain limited to 8 KB.

QMF continues to support a query size of 32 KB for prompted and QBE queries, unless the database to which the RUN QUERY command is directed does not support SQL statements of this size.

### Improved report performance and resource control

Report performance and resource control enhancements include:

- 64-bit virtual storage area for spill data in QMF for TSO

A new 64-bit virtual storage area handles QMF for TSO data that is no longer needed in active storage. This new storage area precludes the need to allocate a spill file in TSO and enables broader support for data types that require larger amounts of storage, such as XML.

A new TSO program parameter, DSQSPTYP, activates this support; a new global variable, DSQEC\_EXTND\_STG, allows you to specify the amount of extended storage for QMF to acquire.

- Increased maximum length for data rows in QMF reports

Prior releases of QMF for TSO and CICS had a 32 KB limit on the length of a single data row returned in a QMF report. A new global variable, DSQEC\_TWO\_GB\_ROW, allows you to set support for row lengths up to 2 GB, with the following restrictions:

- You cannot create a table with a maximum record size that is greater than the page size. Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size. Thus, if you are retrieving data from one table only, the row length is still limited to 32 KB.
- When a table contains a column with LOB data, the data in the LOB column is truncated at 32 KB in query results as well as during display and print operations. The row containing the LOB data can still be up to 2 GB long, however.

The maximum row length remains at 32 KB when you export or import data in QMF, IXF, or HTML format. However, you can use XML format to export or import character data; this format supports record lengths of up to 2 GB.

- Independent governing of database activity generated by user commands

The QMF packages have been restructured so that modules that handle database activity driven by end-user commands are now separate from modules that handle database activity driven by SQL internal to QMF. This new structure allows you to isolate the following groups of QMF commands for individual governing by the Db2 resource limit facility:

- RUN QUERY (SELECT queries of any type), DISPLAY TABLE (when the CONFIRM option is YES), EXPORT TABLE, PRINT TABLE, BOTTOM, TOP, FORWARD, BACKWARD, RIGHT, and LEFT
- SAVE DATA and IMPORT TABLE
- RUN QUERY (when the query includes INSERT, UPDATE, or DELETE statements)
- EDIT TABLE (Add mode)
- EDIT TABLE (Change mode when the SAVE parameter has been set to IMMEDIATE)
- EDIT TABLE (Change mode when the SAVE parameter has been set to END)
- ERASE

- Ability to set concurrent access resolution options from within QMF

This enhancement adds the DSQEC\_CON\_ACC\_RES global variable, which allows you to specify how QMF should resolve locks on data that it is attempting to access. Values for the new DB2 Version 10 bind option, CONCURRENTACCESSRESOLUTION, are supported through this global variable, as is the existing SKIP LOCKED DATA option.

### **Enhancements to QMF commands and support of SQL statements**

Enhancements to QMF commands and support of SQL statements include:

- Optional confirmation prompting with the DISPLAY TABLE command

QMF now provides a CONFIRM parameter for the DISPLAY TABLE command. When CONFIRM is set to YES and the estimated resource to complete the command exceeds the allocated resource defined in the Db2 resource limit facility, QMF displays a panel prompting the user to decide whether to cancel the command.

- Support for implicit casting on the SAVE and IMPORT commands

If the database to which the SAVE or IMPORT command is directed offers support for implicit casting, QMF no longer requires that the columns in the data to be saved or imported have the same data types and lengths as the columns in the existing table that is being replaced or appended.

- Optional default for the OWNER parameter of the QMF LIST command

This enhancement adds a new global variable, DSQEC\_LIST\_OWNER. This variable allows you to specify a default database authorization ID for the OWNER parameter of the LIST command, eliminating the need to specify this parameter each time the command is issued. You can set the variable to any valid database authorization ID, or specify ALL to list all objects that you are allowed to access, regardless of owner. If no value is specified for the variable, the OWNER parameter defaults to the current database authorization ID, as in prior QMF releases.

- Ability to restrict which commands cause the Last Used timestamp on an object to be updated

In previous releases, the timestamp in the LAST\_USED column of the Q.OBJECT\_DIRECTORY table in the QMF catalog was updated whenever an object was referenced by any of the following commands:

- CONVERT
- DISPLAY
- EXPORT
- IMPORT

- LAYOUT
- PRINT
- RUN
- SAVE

This timestamp is displayed in the **Last Used** field on QMF database object lists that are generated by the LIST command. This feature provides a new global variable, DSQEC\_LAST\_RUN, which allows you to default to the timestamp behavior of prior releases or restrict the timestamp to updates by RUN, SAVE, and IMPORT commands only.

- More parameters on SQL CALL statements

This enhancement increases the number of parameters that QMF allows on an SQL CALL statement from 32 to 63.

- Additional support for Db2 special registers

This enhancement allows you to use the SET statement in a QMF SQL query to set the CURRENT SCHEMA and CURRENT REFRESH AGE special registers in Db2 for z/OS.

### Broader support for data types

Enhancements to QMF's support for Db2 data types include:

- Increased support for XML data

Rows that contain XML data can now be up to 2 GB. This new limit is also supported when you import or export data in XML format.

- Broadened support for decimal floating-point data

QMF has been enhanced to provide full support for the decimal floating-point (DECFLOAT) data type, provided that the processor on which QMF is running supports decimal floating-point instructions. QMF supports the long and extended formats of the DECFLOAT data type. If the processor on which QMF is running does not support decimal floating-point instructions, the edit code used for the data defaults to M and metadata is displayed in the report. For example, DECFLOAT(16) is displayed in place of long-format values and DECFLOAT(34) is displayed in place of extended-format values.

- Support for increased precision for the TIMESTAMP data type

This enhancement supports greater precision allowed by Db2 for z/OS Version 10 for the TIMESTAMP data type. The number of digits representing fractional seconds can now be from 0 to 12.

- Support for the new TIMESTAMP WITH TIME ZONE data type

This enhancement provides QMF support for the new DB2 Version 10 data type TIMESTAMP WITH TIME ZONE. This new data type extends the TIMESTAMP data type to include the time zone information. The time zone is the time difference, in hours and minutes, between the local time and Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time (GMT).

### Improvements in QMF forms

This enhancement provides a new global variable, DSQDC\_COL\_LABELS, which controls whether the column heading shown in QMF forms defaults to the database label assigned to the column or the name of the column in the table from which it was selected. Database labels are used by default.

### Improved diagnostic capabilities

Additional information is now provided for QMF errors associated with SQL codes issued by the database manager. For these errors:

- The SQL Communications Area (SQLCA) is now displayed at the end of the QMF message help panel that describes the error.
- A new global variable, DSQDC\_POS\_SQLCODE, allows you to set notification of positive SQL codes at varying levels of detail.

## Version 9.1 changes

---

These are enhancements to QMF Version 9.1.

### Database processing enhancements

The DSQSMRFI program parameter allows you to take advantage of Db2 multiple-row fetch and insert processing. Prior to this change, an SQL FETCH or INSERT statement was required for every row of data processed. With multiple-row processing, a single SQL statement can process several rows of data. For distributed databases, this improvement results in decreased network traffic. QMF can use multiple-row support when the database that is accessed also supports multiple-row fetch or insert. The following QMF commands take advantage of multiple-row support:

BOTTOM  
FORWARD  
DISPLAY TABLE  
DPRE  
EXPORT DATA  
EXPORT TABLE  
PRINT REPORT  
PRINT TABLE  
RUN QUERY  
SAVE DATA.

### Support for Db2 for z/OS Version 9 data types

- Big integer

BIGINT is a numeric data type capable of representing 63-bit integers. The range of numbers supported by BIGINT is -9223372036854775808 to +9223372036854775807.

- Fixed-length and variable-length binary

QMF supports BINARY and VARBINARY data types, which are compatible with the BLOB data type, for which support already existed prior to Version 9.

- XML

Columns in a relational table can contain XML documents. XML columns in Db2 have no architectural limit and no defined length. In Version 9, QMF for TSO and CICS had a size limit of 32 KB for each instance of an XML document. Character data in the XML document is encoded in UTF-8 format. QMF maintains this format when exporting XML data. XML data is converted to character data for viewing in a displayed or printed report.

QMF supports operations with XML data only when you are connected to a database release that supports the XML data type.

- Decimal floating-point

If the processor on which QMF is running supports decimal floating-point instructions, the decimal floating-point data type is fully supported in QMF Version 9 by QMF for WebSphere and QMF for TSO and CICS. If the processor does not support decimal floating-point instructions, QMF for TSO and CICS Version 9 tolerates access to decimal floating-point data by allowing the use of the metadata (M) edit code when displaying a table that contains decimal floating-point data.

### Import/export enhancements

- Ability to import and export XML data

QMF provides a sample style sheet that maps a Db2 table to an HTML table. You can export XML data directly to a z/OS UNIX path or import XML data from a z/OS UNIX path.

- Ability to export and import data using a UNIX path name

This feature provides the ability to export QMF reports in HTML format or export XML data to a Web server, as well as the capability to import XML data from a Web server using a z/OS UNIX path to

access the data. You can map the UNIX path to a file name in a hierarchical file system (HFS) or a file name in a network file system (NFS).

- Ability to export and import QMF objects using PDSE data sets

With this enhancement, you can export and import QMF objects using PDSE data sets. You can also create new PDSE data sets when exporting a QMF object to a new data set. Allocation of PDSE data sets is controlled by the DSQEC\_PO global variable.

- Easier specification of data set size

This enhancement allows you to use a set of QMF global variables to specify the size of a data set used by the Db2 QMF EXPORT command. These variables are DSQEC\_DSALLOC\_PRI, DSQEC\_DSALLOC\_SEC, and DSQEC\_DSALLOC\_DIR.

### **Security enhancements**

The QMF CONNECT command supports the use of RACF mixed-case passwords.

### **Diagnostic capabilities**

This enhancement provides diagnostic support for SQL codes that were new in Db2 for z/OS Version 9.

## **Version 8.1 changes**

---

These are enhancements to QMF Version 8.1.

### **Installation**

The installation process improved in QMF Version 8.1, with less post-SMP/E work to perform.

### **Database operations**

- Names up to 128 characters in length are supported for authorization IDs, current SQL IDs, and table names. Column names can be up to 30 characters long. Support is based on whatever length the database allows.

Support includes larger data entry fields and the display of names in QMF dialog panels.

- The CALL statement can be issued from the **SQL Query** panel to run a Db2 stored procedure. Output parameters are placed in QMF substitution variables defined by the user. Result sets returned have all the features of a result set returned from a query.
- A new global variable, DSQEC\_SP\_RS\_NUM, allows for the specification of a particular result set of a Db2 stored procedure that returns multiple result sets.
- Fully integrated support for BLOB, CLOB, and DBCLOB data types. LOB data can be displayed in a QMF report or printed from either temporary storage or the database, but is truncated at 32 KB.

The M edit code is assigned to all LOB columns by default. This edit code displays the associated column metadata in lieu of the actual data. Other edit codes can be used, depending on the data type.

- Support for several new Db2 UDB for z/OS Version 8 SQL codes.
- As of QMF Version 8.1, only Db2 UDB for z/OS databases can function as requesters. Additionally, the QMF runtime libraries are provided for z/OS only, not for VM or VSE. This means that QMF can only be started in a Db2 UDB for z/OS database. QMF Version 8.1 running on a Db2 UDB for z/OS database can still connect to a server that is not running z/OS as long as QMF Version 8.1 is installed on that server.

### **QMF commands**

- A new PRINT command option allows the suppression of carriage control characters in the report output format when a print device name is not supplied.

The DSQEC\_CC global variable provides flexibility in setting the CC (carriage control) keyword on the PRINT command.



- The size of global variables specified on the SET GLOBAL and GET GLOBAL commands was extended in QMF Version 8.1 from 55 to 2000 characters.

## Version 7.2 changes

---

These are enhancements to QMF Version 7.2.

### Installation

- Db2 UDB for OS/390® and z/OS environments included the user exit library QMF720.SDSQEXIT. This library can be used to store the user-modified version of the DSQUOPTS routine, local user date and time exits (Db2-supplied exits), QMF user edit code routines, and QMF callable interface programs.
- OS/390 and VM environments included the DSQ1BVW installation job, which creates or re-creates all default QMF views on any supported Db2 database. Installation jobs were also provided for the enhanced LIST command function, which allows the QMF LIST command to list all tables and views authorized to either a primary or secondary authorization ID so that you do not have to explicitly grant privileges to PUBLIC.

### Database operations

- QMF Version 7.2 introduced the ability for QMF on VM, VSE, and z/OS to connect to Db2 for Linux/390 as an application server.
- Support was added for several new Db2 UDB for OS/390 and z/OS SQL codes.

### QMF initialization

- The DSQAO\_QMFADM global variable was introduced. This variable indicates whether the user who started QMF is a QMF administrator. Global variable DSQEC\_DISABLEADM allows you to turn this checking off.
- The DSQUOPTS exit routine was made available to override default values for global variables DSQEC\_SHARE and DSQEC\_DISABLEADM.

### Application support

The @IF REXX function was made available for FORM.CALC, FORM.CONDITIONS, and FORM.COLUMNS definitions to provide an extended ability to conditionally alter the display of data with improved null, overflow, and undefined processing. This spreadsheet-like function provides QMF report authors with extended control over report output based on input values.

### QMF commands and object operations

- The QMF LIST TABLES command works with authority groups defined by Db2 secondary authorization IDs.
- Support was introduced for the QMF CONNECT command to enable uniform support of Db2 for OS/390 user IDs and passwords across three QMF platforms: VM, VSE and OS/390.
- Users are prompted before discarding modifications in QUERY, PROC, FORM, and PROFILE objects. This function is provided by the following global variables:
  - DSQEC\_PRO\_QUERY
  - DSQEC\_PRO\_PROC
  - DSQEC\_PRO\_FORM
  - DSQEC\_PRO\_PROF
  - DSQEC\_PRO\_ENABLE
- The SET GLOBAL command allows procedure authors to set the value of a global variable to the value of another global variable.

## Version 7.1 changes

---

These are enhancements to QMF Version 7.1.

QMF Version 7.1 provided these enhancements.

### Database operations

- Distributed access to the entire Db2 family of server products was introduced in this release, with the addition of support for:
  - DB2 for AS/400 server, Version 4.4
  - DB2 for VSE Application Requester
- Cross-platform DRDA package binding provided additional installation and maintenance options for QMF on VM, VSE and OS/390.
- Support for the following data types:
  - ROWID data type  
QMF Version 7.1 provided fully integrated support for this data type.
  - LOB data types  
QMF Version 7.1 provided limited support for this data type in the following areas:
    - Table Editor
    - Prompted Query
    - DRAW SQL Query

QMF restricts LOB lengths to 32 KB in QMF reports. New SQL functions cast LOB columns into character columns.

### QMF commands and object operations

- Date and time edit codes TD and TT were added. These codes adjust their report formatting characteristics automatically based on changes in the date or time format of the database management system.
- Defaults were implemented for QMF commands that draw their values from the current context. Working with the PROC, QUERY, or FORM objects was therefore made easier for the following commands:
  - CONVERT
  - EDIT
  - EXPORT
  - PRINT
  - RESET
  - RUN
  - SAVE
- Additional flexibility and control was provided for command options that accept quoted strings, including more ways to specify the string. Increased lengths were also offered. Featured options include:
  - COMMENT for the SAVE and IMPORT commands
  - TEXT for the MESSAGE command
  - FROM for the CICS command (transaction data)
  - Direct navigation to the QMF home panel using the SHOW command.

## Version 6.1 changes

---

These are enhancements to QMF Version 6.1.

### **Installation**

Support for the ISPF LIBDEF service was introduced. You can use this service to allocate QMF program libraries.

### **Database operations**

- Support for Db2 Universal Database Version 5 was introduced.
- Formal QMF administrator authority was introduced, which provides administrators with the unrestricted ability to display, save, erase, import, and export QMF objects that are owned by any user.
- Support for Db2 special registers was added.

### **QMF initialization**

The user's current authorization ID is displayed prominently on the QMF home panel.

### **Application support**

Support for developing user-defined edit codes using Language Environment was added.

### **Resource control**

Users can choose in advance whether to run or cancel a long-running query that is expected to use too many resources. This function is provided by a confirmation panel that works with the Db2 predictive governor.

### **QMF commands and object operations**

- The following improvements were introduced in QMF forms and reports:
  - Support for the euro currency symbol.
  - An edit code that offers suppression of zeroes by replacing zeroes with blanks in QMF reports.
  - An edit code and associated global variable for currency formatting, enabling two different currency symbols in the same report.
  - A global variable that sets the default for the QMF scroll amount.
- The following improvements were introduced in QMF commands:
  - The ERASE TABLE command was introduced, which can be used to erase a database alias.
  - Improvements to SAVE command syntax to allow QMF to recognize both the object name and type in context.
- Table editor support for column defaults in Change mode was added



## Appendix B. QMF functions that require specific support

Support for these functions varies with the database or environment.

### Functions that vary according to database type

Support for these functions varies by database.

*Table 82. Functions that require the support of specific database management systems*

| Function supported                                                     | Db2 for z/OS                                                                                                                                                                 | Db2 for Linux, UNIX, and Windows                                                                                    | DB2 for iSeries                                                                                                     | DB2 for VSE and VM                                                       |
|------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| Length of query statement supported                                    | 2 MB*                                                                                                                                                                        | 65 KB*                                                                                                              | 65 KB*                                                                                                              | 8 KB                                                                     |
| Number of columns in SELECT statement                                  | 750                                                                                                                                                                          | 255                                                                                                                 | 255                                                                                                                 | 255                                                                      |
| Importing single-precision floating point numbers                      | X                                                                                                                                                                            |                                                                                                                     |                                                                                                                     | X                                                                        |
| Long fields with LIKE statement                                        | X                                                                                                                                                                            |                                                                                                                     |                                                                                                                     | X                                                                        |
| Database synonyms                                                      |                                                                                                                                                                              |                                                                                                                     |                                                                                                                     | X                                                                        |
| Database aliases for tables or views                                   | X                                                                                                                                                                            | X                                                                                                                   | X                                                                                                                   |                                                                          |
| SAVE=IMMEDIATE option available in Table Editor (supports CURSOR HOLD) | X                                                                                                                                                                            | X                                                                                                                   | X                                                                                                                   |                                                                          |
| Setting Db2 global variables                                           | X                                                                                                                                                                            | X                                                                                                                   |                                                                                                                     |                                                                          |
| QMF commands that include three-part names                             | Commands with three-part names can be initiated from this type of database. They can also be directed to this type of database unless QMF was started as a stored procedure. | Commands with three-part names can be directed to this type of server unless QMF was started as a stored procedure. | Commands with three-part names can be directed to this type of server unless QMF was started as a stored procedure. | Commands with three-part names cannot be directed to these server types. |

\* To activate support for SQL queries up to 2 MB on Db2 for z/OS databases and up to 65 KB on Db2 for Linux, UNIX, and Windows databases, set the DSQEC\_SQLQRYSZ\_2M global variable to 1 before running the query.

#### Related concepts

[Starting QMF as a Db2 for z/OS stored procedure](#)

The stored procedure interface to QMF for TSO allows any software program that can call a Db2 for z/OS stored procedure to start QMF for TSO. The interface can also run a predefined QMF query or procedure and receive up to 21 result sets back, including one for trace data.

## Functions not available in CICS

---

Certain functions are supported by TSO only.

The following functions are not available in CICS:

- Use of multiple thread support.
- Use of QMF Analytics for TSO
- Use of the QMF Enhanced Editor.
- Use of extended storage for spilling report data no longer needed in active storage; a spill file must be used instead
- Ability to start QMF as a Db2 for z/OS stored procedure
- Interfaces:
  - Command interface
  - Document interface
- Program parameters:
  - DSQSCMD (QMF callable interface only)
  - DSQSMTHD
  - DSQSPLAN
  - DSQSPRID
  - DSQSPTYP
  - DSQSRSTG
  - DSQSSUBS
- Commands:
  - BATCH (and its associated application)
  - DPRE (and its associated REXX exec)
  - EDIT QUERY
  - EDIT PROC
  - ISPF (and its associated application)
  - LAYOUT (and its associated application)
  - SET GLOBAL commands that reference the following global variable:
    - DSQEC\_EXTND\_STG
  - SHOW GLOBAL commands that reference the following global variables:
    - DSQEC\_EXTND\_STG
    - DSQAO\_STO\_PROC\_INT
  - STATE (requires the command interface)
- Macros: GETQMF
- Form functions:
  - Report calculations or expressions that require REXX
  - Conditional formatting
  - Column definition

- Locally defined edit codes TDL and TTL (for formatting dates and times, respectively)
- Procedures with logic (which require REXX)
- The ability to cancel transactions
- The ability to update data at remote locations (all tables and views at remote locations are read-only in QMF for CICS)
- External variables





## Appendix C. QMF objects that reside in Db2

These QMF objects are necessary to run QMF Version 12.1 in a Db2 for z/OS subsystem. You can use this information as a guide during recovery operations if necessary.

### QMF plans

These plans are shipped with Db2 QMF for TSO and CICS.

**Important:** Do not change or add any BIND PACKAGE or BIND PLAN options in any of the QMF bind jobs for plans and installation packages unless instructed to do so in this information, in the job itself, or by IBM Software Support.

| Plan name | Bind job that installs this plan | Notes                                    |
|-----------|----------------------------------|------------------------------------------|
| QMF1210   | DSQ1BINR                         | Default QMF plan                         |
| DSQIN121  | DSQ1BSQL                         | QMF plan used for installation jobs only |

### QMF packages

These packages are shipped with QMF.

| Package name              | Bind job     |
|---------------------------|--------------|
| DSQJ*                     | DSQ1BPKG JCL |
| For remote servers: DSQJ* | DSQ1BPKG JCL |

### QMF control tables and table spaces for TSO and CICS

These are the control tables shipped with QMF.

If you installed QMF in a DB2 for iSeries database, these control tables are installed in collection Q instead of in these table spaces.

**Important:** To ensure that you can recover QMF operations to a specific point in time if necessary, back up all of these tables at the same time.

| Control table name | Table space | Table space size (in 1 KB units) | Table content                                                                                                          | Index               |
|--------------------|-------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------|---------------------|
| Q.PROFILES         | DSQTSPRO    | 100 primary, 20 secondary        | Contains QMF profiles that hold information about individual users' access to resources and data during a QMF session. | Q.PROFILEX          |
| Q.OBJECT_DIRECTORY | DSQTSCT1    | 200 primary, 20 secondary        | Contains general information about all QMF queries, forms, procedures, folders, and analytic objects in the database.  | Q.OBJECT_DIRECTORYX |

Table 85. QMF control tables and the table spaces in which they reside (continued)

| Control table name              | Table space | Table space size (in 1 KB units) | Table content                                                                                                                                                                                                                                                                                                                                  | Index                 |
|---------------------------------|-------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| Q.OBJECT_REMARKS                | DSQTSCT2    | 200 primary, 20 secondary        | Contains comments that were saved when queries, forms, procedures, and analytics objects were created or replaced.                                                                                                                                                                                                                             | Q.OBJECT_REMARKSX     |
| Q.OBJECT_DATA<br>Q.OBJECT_DATA2 | DSQTSCT3    | 5000 primary, 200 secondary      | Contains queries, forms, procedures, folders, and analytic objects represented in a format that is internal to QMF. Q.OBJECT_DATA2 is for Db2 for Linux, UNIX, and Windows databases only.                                                                                                                                                     | Q.OBJECT_OBJDATA      |
| Q.COMMAND_SYNONYMS              | DSQTSSYN    | 100 primary, 20 secondary        | Contains information about QMF command synonyms. The Q.COMMAND_SYNONYMS table stores synonyms for English.                                                                                                                                                                                                                                     | Q.COMMAND_SYNONYMNSX  |
| Q.COMMAND_SYNONYM_n             | DSQTSSYn    | 100 primary, 20 secondary        | Contains information about the QMF command synonyms. The Q.COMMAND_SYNONYMS table stores synonyms for English. There is a Q.COMMAND_SYNONYM_n table for each QMF NLF; replace the n with the 1-character identifier for the national language you are using. For older QMF installations, these tables may all be in the DSQTSSYN table space. | Q.COMMAND_SYNONYMNX_n |
| Q.RESOURCE_TABLE                | DSQTSGOV    | 100 primary, 20 secondary        | Contains resource control information passed to the governor exit routine.                                                                                                                                                                                                                                                                     | Q.RESOURCE_INDEX      |
| Q.ERROR_LOG                     | DSQTSLOG    | 100 primary, 20 secondary        | Contains information about system, resource, and unexpected errors. This information is more detailed than the information provided in error messages.                                                                                                                                                                                         | None                  |
| Q.DSQ.RESERVED                  | DSQTSRDO    | 12 primary, 4 secondary          | Contains information used by QMF during installation.<br><b>Important:</b> Do not modify this table.                                                                                                                                                                                                                                           | None                  |
| Q.GLOBAL_VARS                   | DSQTSGLV    | 110 primary, 20 secondary        | Contains system and user-defined global and session variables.                                                                                                                                                                                                                                                                                 | Q.GLOBAL_VARSX        |

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

Installation path G: [Preparing a remote server to be accessed by QMF commands that include three-part names](#)

## QMF views

These views are shipped with QMF. QMF uses these views on the platforms indicated in the table to create object lists when the QMF LIST command is issued.

Table 86. Views used to create object lists when the QMF LIST command is issued

| View name         | Table or tables on which the view is based on each platform                                                                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Q.DSQ_GLOBAL_VARS | Q.GLOBAL_VARS (for all platforms). This view is not created on DB2 for VSE and VM.                                                                                                                          |
| Q.DSQ_RESERVED_DB | <ul style="list-style-type: none"> <li>• SYSIBM.SYSCOLUMNS (for Db2 for z/OS)</li> <li>• SYSCAT.COLUMNS (for Db2 for Linux, UNIX, and Windows)</li> <li>• QSYS2.SYSCOLUMNS (for DB2 for iSeries)</li> </ul> |

Table 86. Views used to create object lists when the QMF LIST command is issued (continued)

| View name          | Table or tables on which the view is based on each platform                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Q.DSQ_RESERVED_OBJ | Q.OBJECT_DIRECTORY (for all platforms)                                                                                                                                                                                                               |
| Q.DSQEC_ALIASESL   | <ul style="list-style-type: none"> <li>• SYSIBM.SYSTABLES (for Db2 for z/OS)</li> <li>• SYSCAT.TABLES (for Db2 for Linux, UNIX, and Windows)</li> <li>• QSYS2.SYSTABLES (for DB2 for iSeries)</li> </ul>                                             |
| Q.DSQEC_COLS_LDB2L | <ul style="list-style-type: none"> <li>• SYSIBM.SYSCOLUMNS and SYSIBM.SYSTABAUTH (for Db2 for z/OS)</li> <li>• SYSCAT.COLUMNS and SYSCAT.TABAUTH (for Db2 for Linux, UNIX, and Windows)</li> <li>• QSYS2.SYSCOLUMNS (for DB2 for iSeries)</li> </ul> |
| Q.DSQEC_COLS_RDB2L | SYSIBM.SYSCOLUMNS and SYSIBM.SYSTABAUTH (for Db2 for z/OS)                                                                                                                                                                                           |
| Q.DSQEC_QMFOBJSL   | Q.OBJECT_DIRECTORY and Q.OBJECT_REMARKS (for all platforms)                                                                                                                                                                                          |
| Q.DSQEC_TABS_LDB2L | <ul style="list-style-type: none"> <li>• SYSIBM.SYSTABAUTH and SYSIBM.SYSTABLES (for Db2 for z/OS)</li> <li>• SYSCAT.TABAUTH and SYSCAT.TABLES (for Db2 for Linux, UNIX, and Windows)</li> <li>• QSYS2.SYSTABLES (for DB2 for iSeries)</li> </ul>    |
| Q.DSQEC_TABS_RDB2L | SYSIBM.SYSTABAUTH and SYSIBM.SYSTABLES (for Db2 for z/OS)                                                                                                                                                                                            |
| Q.RESOURCE_VIEW    | Q.RESOURCE_TABLE (for all platforms)                                                                                                                                                                                                                 |

Several of these views are based on Db2 system tables and are used by QMF for both the LIST and DESCRIBE functions.

You can create or re-create all default QMF control table views on any supported Db2 database from z/OS by running job DSQ1BVW. This job drops any existing views, creates new views, and grants the necessary privileges on the views. If you installed the enhanced LIST command function, which is available for use with Db2 for z/OS, run job DSQ1BUDV to install the views after you run job DSQ1BVW. Job DSQ1BUDV enables QMF control table views for Db2 secondary authorization IDs.

**Note:** Depending on the value of installation parameter EXTSEC, the definitions of views Q.DSQEC\_TABS\_LDB2L, Q.DSQEC\_COLS\_LDB2L, Q.DSQEC\_TABS\_RDB2L and Q.DSQEC\_COLS\_RDB2L may not include reference to SYSIBM.SYSTABAUTH.

### Related tasks

Installing the enhanced LIST command function (z/OS only)

When the enhanced LIST command function is installed, privileges for tables and views must be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be included in object lists that are generated by the LIST TABLES or LIST ALL command when the OWNER=ALL option is specified on the command.

## Db2 for z/OS storage groups

These Db2 for z/OS storage group (STOGROUP) objects are created during QMF installation.

| Storage group name | Description                                                       |
|--------------------|-------------------------------------------------------------------|
| DSQ1STBG           | Stores QMF sample tables                                          |
| DSQSGCTL           | Stores QMF control tables                                         |
| DSQSGDEF           | Default storage group for tables created by the SAVE DATA command |

Table 87. Db2 for z/OS storage groups that are created for QMF (continued)

| Storage group name | Description                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DSQSGSY $n$        | Stores the command synonym index for QMF national language features (Q.COMMAND_SYNONYMNSX)<br><br>There is a separate space for each national language, indicated by a unique character in place of the $n$ . |

Do not configure these storage groups as user-managed groups after the QMF installation.

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Space for saving LOB data

When you issue a SAVE DATA command for a table that contains one or more LOB columns, multiple objects are created by Db2 and QMF.

QMF creates one base table and index in the table space that is specified in the SPACE field of the QMF profile.

Db2 creates an auxiliary table space, table, and index for each LOB column in the table that is saved.

## VSAM clusters for TSO/CICS

These VSAM clusters are shipped with QMF.

| Cluster name                                | Object for which cluster is needed |
|---------------------------------------------|------------------------------------|
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSTCT1.I0001.A001 | DSQTSTCT1                          |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSTCT2.I0001.A001 | DSQTSTCT2                          |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSTCT3.I0001.A001 | DSQTSTCT3                          |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSPRO.I0001.A001  | DSQTSPRO                           |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSLLOG.I0001.A001 | DSQTSLLOG                          |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSGOV.I0001.A001  | DSQTSGOV                           |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSSYN.I0001.A001  | DSQTSSYN                           |
| QMFDSN.DSNDBC.DSQDBCTL.DSQTSGLV.I0001.A001  | DSQTSGLV                           |
| QMFDSN.DSNDBC.DSQDBCTL.OBJECTRD.I0001.A001  | Q.OBJECT_DIRECTORYX                |
| QMFDSN.DSNDBC.DSQDBCTL.OBJECTRR.I0001.A001  | Q.OBJECT_REMARKSX                  |
| QMFDSN.DSNDBC.DSQDBCTL.OBJECTRO.I0001.A001  | Q.OBJECT_OBJDATA                   |
| QMFDSN.DSNDBC.DSQDBCTL.PROFILEX.I0001.A001  | Q.PROFILEX                         |
| QMFDSN.DSNDBC.DSQDBCTL.COMMANDR.I0001.A001  | Q.COMMAND_SYNONYMSX                |
| QMFDSN.DSNDBC.DSQDBCTL.GLOBALRV.I0001.A001  | Q.GLOBAL_VARSX                     |

## QMF sample tables

QMF provides sample tables that you can use to help you learn and test product functions. These tables contain data about applicants, interviews, parts, products, employees, and suppliers of a fictitious electrical parts manufacturer, J & H Supply Company.

| Table       | Db2 for z/OS Table Space Name | Contains information about:            |
|-------------|-------------------------------|----------------------------------------|
| Q.APPLICANT | DSQ1ST3E or DSQ1STBT          | New candidates for hire                |
| Q.INTERVIEW | DSQ1ST6E or DSQ1STBT          | Interviews of new hires                |
| Q.ORG       | DSQ1ST1E or DSQ1STBT          | The company's organizational structure |
| Q.PARTS     | DSQ1ST9E or DSQ1STBT          | Product parts data                     |
| Q.PRODUCTS  | DSQ1ST4E or DSQ1STBT          | The company's products                 |
| Q.PROJECT   | DSQ1ST7E or DSQ1STBT          | Projects undertaken, by department     |
| Q.SALES     | DSQ1ST5E or DSQ1STBT          | Sales and commissions                  |
| Q.STAFF     | DSQ1ST2E or DSQ1STBT          | The company's personnel                |
| Q.SUPPLIER  | DSQ1ST8E or DSQ1STBT          | Vendor information                     |

Additionally, QMF Analytics for TSO provides sample tables that you can use to learn about QMF Analytics for TSO functions. The following table shows the type of data that is contained in each sample table.

| Table          | Db2 for z/OS Table Space Name | Contains information about:                                                                            |
|----------------|-------------------------------|--------------------------------------------------------------------------------------------------------|
| Q.CASHFLOW     | DSQ1STAE or DSQ1STBT          | Data about cost and revenue that can be used with QMF Analytics for TSO discounted cash flow analytics |
| Q.CLIMATE_10YR | DSQ1STAE or DSQ1STBT          | Data about climate over a 10-year period                                                               |
| Q.CLIMATE_USA  | DSQ1STCE or DSQ1STBT          | Data about climate in the United States, including data about rainfall and sunshine                    |
| Q.WORLDINFO    | DSQ1STBE or DSQ1STBT          | Data about geographic regions in which the J & H Supply Company conducts business                      |

In DB2 for Linux, UNIX, and Windows databases, the DSQ1STBT table space is found in the DSQTSAMP database partition group. The sample tables reside in the DSQ1STBT table space. In DB2 for z/OS databases, the table spaces listed in the charts reside in the DSQ1STBB database.



---

## Appendix D. External editors

You can use outside editors with QMF objects and reports.

---

### Modifying QMF queries and procedures with an editor

---

You can use the EDIT command to modify a QMF SQL query or procedure that is in temporary storage by using an editor external to QMF.

#### Before you begin

The following procedure assumes that you use an editor that can be called by a CLIST operating under ISPF. However, if the editor is ISPF/PDF, it is called directly using the ISPF EDIT service instead of through a CLIST

#### About this task

One of the external editors can be ISPF/PDF (if QMF is started under ISPF). When you issue the EDIT command, QMF exports the object from QMF temporary storage to a temporary edit transfer data set. When the edit session is complete, the contents of the temporary data set are imported into temporary storage.

#### Procedure

To make an editor available for the EDIT command:

1. Write a CLIST to call the editor and pass the name of the data set to be edited as a positional parameter.  
For example, with the following command, QMF calls a CLIST named XYZEDIT to edit a data set named USERA.XYZDATA.TEXT:

```
XYZEDIT 'USERA.XYZDATA.TEXT'
```

2. Place the CLIST in a command library that is allocated to everyone with access to the editor. Place it in a library that is part of the concatenation for the ddname SYSPROC. One possible choice is QMF library QMF1210.SDSQCLTE, which must be available to all QMF users.
3. For individual users, allocate and catalog a data set for objects to be edited.  
This data set is refilled every time that the user calls the editor with the EDIT command. Give the data set the following characteristics:

- A physical sequential organization (DSORG=PS)
- Fixed-length, 79-byte records (LRECL=79)
- A block size of 4029 (BLKSIZE=4029)

4. In the JCL for each user, allocate the data set that is cataloged for that user in step “3” on page 393. Allocate it with the ddname DSQEDIT. Use DISP=OLD for the disposition of the data set.

5. Tell users how to specify the EDIT command.

The command has the following format:

```
EDIT yyyy (EDITOR=xxxx)
```

In this syntax, *yyyy* is either PROC or QUERY, and *xxxx* is the name of the CLIST created to call the editor.

6. You can edit your QMF SQL query or QMF procedure under a different ISPF application ID by using an exec or CLIST as the editor name on the QMF EDIT command.

- If you specify the program development facility (PDF) editor to edit an SQL query or QMF procedure, QMF executes the PDF editor under the QMF application ID DSQE, or DSQ $n$  where  $n$  is a 1-character language identifier. In addition, QMF sets the function keys and location of the command line to fit the QMF product.
- If you must use a different set of function keys or have existing PDF macros or specialized PDF editor screens, you can use them by executing the PDF editor under an application ID other than DSQ $n$ . Execute two small REXX programs or CLISTs. The first program simply routes execution to the second program. The second program then invokes the editor that is running under the ISPF application ID with the function key that you want, or with other special setup requirements such as an edit invocation macro or a unique edit panel.

This REXX program example shows how to edit the SQL query or QMF procedure by using the edit transfer data set, as defined by ddname(DSQEDIT), when QMF is started. The PDF application ID ISP is used in this example.

```

Edit Program 1 (MYEDIT)

/* REXX   QMF Edit program 1           */
/*       Transfer to ISP application ID */
Address ISPEXEC "SELECT CMD(MYEDIT2) NEWAPPL(ISP)"
Exit 0

Edit Program 2 (MYEDIT2)

/* REXX   QMF Edit program 2           */
/*       Invoke PDF Editor using DDNAME */
Address ISPEXEC "LMINIT DATAID(EDT) DDNAME(DSQEDIT)"
Address ISPEXEC "EDIT  DATAID("EDT")"
Address ISPEXEC "LMFREE DATAID("EDT")"
Exit 0

```

Figure 126. Editing by using the edit transfer data set

The REXX programs must be allocated to a valid concatenation of either SYSPROC or SYSEXEC before execution. To execute from QMF, issue a command like the following QMF EDIT command from the QMF command line:

```
EDIT QUERY (E=MYEDIT)
```

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Inserting QMF reports into documents

The document interface is a macro that is supplied by IBM for the ISPF/PDF and PS/TSO editors; it is not available in CICS. Using this macro, a user operating outside QMF can begin a QMF session and insert a QMF report into a document while the document is being edited.

The report can be created before the editing session begins. More importantly, the user can create the report at the time the GETQMF macro is issued, in a QMF session that the macro started.

Before your users can use this macro, you must:

- Ensure that each user is operating with the proper QMF libraries.

In the sample TSO logon procedure, these libraries have names of the following form:

```
QMF1210.SDSQ*
```



You can operate the ISPF/PDF and PS/TSO editors without these resources; however, the document interface cannot successfully begin a QMF session.

- Change certain document interface components.

Some of these changes are required, while others are optional.

If you are using an NLF, you must also customize the NLF version of the document interface.

## Changing the application

Change the application by changing one or more of its components.

The components that you can change are members of certain QMF libraries:

- The CLISTs and macros are members of QMF1210.SDSQCLTE.
- The other components are members of QMF1210.SDSQSAPE.

## Renaming the document interface macro (DSQAED1P)

The macro component, DSQAED1P, is the macro that users call to use the document interface.

### Procedure

To use the macro:

- Rename a copy of the macro, preferably to GETQMF.  
This default name is used for the macro.

If you are using an NLF, the main macro is member DSQA $n$ D1P of the library QMF1210.SDSQCL $T_n$ , where  $n$  is the 1-character language ID for the NLF you are using. Like the main English-language macro, the macro can be renamed with no effect on the other components. Choose a name other than GETQMF if your users' JCL supports both the English-language and NLF environments. You might consider changing it to GETQMF $n$ , for example.

- Place the renamed copy in the library that contains the original (QMF1210.SDSQCLTE).

### Related tasks

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Placing the Q.DSQAED1S procedure in the database

The Q.DSQAED1S procedure installs the document editing interface function. You place the procedure in the database as part of setting up the document editing interface.

### About this task

The Q.DSQAED1S procedure is in the member DSQAED1S of the QMF1210.SDSQSAPE library.

If you are using an NLF, change the 1-character national language ID in the member DSQA $n$ D1S of the QMF1210.SDSQSAP $n$  library.

### Procedure

- As a QMF administrator, you can place Q.DSQAED1S in the database by entering the following QMF command:

```
IMPORT PROC DSQAED1S FROM 'QMF1210.SDSQSAPE(DSQAED1S)' (SHARE=YES)
```

### Related tasks

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## Changing the data components

Before your users can use this macro to insert QMF reports into documents, you must change certain data components. Some of these changes are required, while others are optional.

There are five data components, all in the library QMF1210.SDSQSAPE. Unlike the CLISTs and macros, these components do not contain logic or executable commands. Instead, they contain information that can appear in messages or in users' reports.

Because the document interface assumes that these components are in a single library, you can modify them in either of the following ways:

- You can retain the changed components in QMF1210.SDSQSAPE.

If you use this method, change the names of the original components, and give the changed components the original names.

- You can place the changed components in a new library.

If you use this method, you must copy all the other data components from the old library into the new library. You must change the macro DSQAED1P.

### The message component

One of the five data components is named DSQAED0L. This component contains messages that can appear on a user's screen while the user is operating the document interface, and keywords for certain QMF commands.

Do not change this component.

If you are using an NLF, change the 1-character language ID in the member DSQA $n$ D0L of the QMF1210.SDSQSAP $n$  library.

### The DCF components

The Document Composition Facility (DCF) is an IBM text processing system. If your site uses DCF, you might want to change the remaining four DCF components.

A user can indicate to the document interface that the current document is formatted by DCF. In response, the document interface adds DCF control statements to the user's inserted report. Wherever these statements appear, they consist of all the records in one or another of the DCF components. You can change any or all of the records in a component. The components, and what they supply, are as follows:

- DSQABD01

Supplies statements that are inserted just before the report. These default statements are in the default component:

```
.* QMF Document Interface heading control:  
.SA  
.RH SUP  
.RF SUP  
.HS 0  
.FS 0  
.TM 0.5I  
.BM 0  
.DC CONT OFF  
.FO OFF
```

- DSQABD02

Supplies statements that are inserted just after each page footing. This default statement is in the default component:

```
. * QMF Document Interface page footing control:
```

- DSQABD03

Supplies statements that are inserted just before each page heading. These default statements are in the default component:

```
. PA NOSTART  
. * QMF Document Interface page heading control:
```

- DSQABD04

Supplies statements that are inserted just after the end of the report. These default statements are in the default component::

```
. * QMF Document Interface footing control:  
. RE  
. * QMF REPORT END
```

### Related tasks

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

### Related information

Search for information about the Document Composition Facility components in your DCF information in the DCF information.

## Changing the CLISTs and macros

To set up the document editing interface, you might need to make more changes to some components.

These components are all in the library QMF1210.SDSQCLTE. If you change the CLISTs or macros, change a copy (not the original) and place it in another library. A DD statement for the new library must appear among the statements for SYSPROC in your users' JCL. If it is not there already, insert one before the statement for QMF1210.SDSQCLTE. Otherwise, the original components are used, instead of the ones you modified. For example, if you place the modified components in the library XYZ.NEWCLIST, then the DD statements for SYSPROC might look like the following statements:

```
//SYSPROC DD DSN=SYSUT2.CLIST,DISP=SHR  
// DD DSN=XYZ.NEWCLIST,DISP=SHR  
// DD DSN=QMF1210.SDSQCLTE,DISP=SHR
```

### Changing DSQAnD1P

This is the macro that you renamed GETQMF. You can also do the following to the macro:

- Change the following statements:

```
SET &SAMPLIB = QMF1210.SDSQSAP&LANGCHAR  
SET &BASELIB = QMF1210.SDSQSAPE
```

#### **&SAMPLIB**

Identifies the library that contains the data components of the document interface

&LANGCHAR refers to the 1-character ID that identifies each NLF.

#### **&BASELIB**

Identifies the QMF samples library, QMF1210.SDSQSAPE

When &LANGCHAR has the value E, both variables name the same library – QMF1210.SDSQSAPE. If the libraries have different names, change the names assigned: &SAMPLIB and &BASELIB.

- Change the following statement:

```
ALLOC FI(DSQPRINT) SYSOUT RECFM(F B A) LRECL(133) BLKSIZE(1330)
```

A user can call the document interface in an interactive QMF session. When called this way, the document interface can reallocate DSQPRINT. This statement restores DSQPRINT to the default. If you do not want DSQPRINT restored to the default, replace this statement with one that restores DSQPRINT to the value you want.

### Changing DSQABD1Q

This CLIST allocates data sets for the session started with the document interface. Make whatever modifications you think are necessary to the CLIST code. For example, you might need to add allocations for data sets specific to your site.

Some of these allocations include GDDM data sets. The document interface does not itself use these data sets, but you might find this allocation necessary.

The variable &LANGCHAR has the value E. This value indicates a library that contains English-language components, as opposed to components for a German application, for example.

To support LIBDEF allocations, activate the LIBDEF service and customize file names as necessary:

```
/* Remove the Following "GOTO NOLIBDEF" statement to allocate */
/* ISPF libraries using the ISPF LIBDEF service. */
/* ***** */
GOTO NOLIBDEF
/* ***** */
/* ALLOCATE QMF ISPF LIBRARIES USING LIBDEF */
/* ***** */
SET PNAME = 'QMF1210.SDSQPLB&LANGCHAR' /* ISPF Panel Library */
SET MNAME = 'QMF1210.SDSQMLB&LANGCHAR' /* ISPF Message Library */
SET SNAME = 'QMF1210.SDSQSLB&LANGCHAR' /* ISPF Skeleton Library */
SET LNAME = 'QMF1210.SDSQLOAD' /* QMF Modules */
ISPEXEC LIBDEF ISPLIB DATASET ID(&PNAME)
```

### Changing DSQABD1P to support LIBDEF

If you allocated QMF libraries with the LIBDEF function, modify DSQABD1P to free the use of LIBDEF-allocated libraries. Uncomment the following statements in DSQABD1P:

```
/* ***** */
/* FREE ISPF LIBDEFs */
/* You might or might not need to free libdefs here. */
/* If you do, then remove comments from LIBDEF statements. */
/* ***** */
/* ISPEXEC LIBDEF ISPLIB DATASET ID() */
/* ISPEXEC LIBDEF ISPLIB DATASET ID() */
/* ISPEXEC LIBDEF ISPLIB DATASET ID() */
/* ISPEXEC LIBDEF ISPLIB DATASET ID() */
/* FREE FI(DSQQLIB) */
```

### Changing DSQABD1C

You can modify this component in the following ways:

- Change the following statement:

```
ALLOC FI(DSQPRINT) UNIT(SYSDA) SPACE(5,2) TRACKS +
RECFM(F B A) LRECL(&PRINTREC) BLKSIZE(&EVAL(&PRINTREC*10))
```

This statement allocates a data set for the user's report. The user then fills the data set through the QMF PRINT command. You might need to change the SPACE operand if your users create large reports.

- Change the following statement:

```
ISPEXEC SELECT PGM(DSQQMF&LANGCHAR)
                PARM(I=&PROCNAME)
                NEWAPPL(DSQ&LANGCHAR)
```

With the statement in its present form, the subsystem for Db2 must be named DSN, and the application plan for QMF must be named QMF12. If your settings are different, you must add information to the PARM operand of the statement. The following example statement applies to a subsystem and application plan named ABC and QMFXXX:

```
ISPEXEC SELECT PGM(DSQQMF&LANGCHAR)
                PARM(I=&PROCNAME , S=ABC , P=QMFXXX)
                NEWAPPL(DSQ&LANGCHAR)
```

The modified statement overrides default values for some QMF program parameters.

### **Related concepts**

#### Setting program parameters and preferences at startup time

This topic describes program parameters that you can pass to QMF when starting QMF under the standard interface.

### **Related tasks**

#### Installing QMF National Language Features

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.



# Appendix E. QMF user-defined functions

Information about QMF user-defined functions.

## QMF user-defined functions and procedures

Some of the optional QMF installation features allow you to create user-defined functions and procedures.

The following table lists the user-defined functions and procedures that are associated with optional installation features. The user-defined functions and procedures that are available in your environment will vary based on the optional features that are installed.

*Table 91. User-defined functions and procedures*

| Name             | Specific name | Type      | Optional installation job that defines the function or procedure | Use                            |
|------------------|---------------|-----------|------------------------------------------------------------------|--------------------------------|
| Q.DSQQMFSP       | DSQQMFSP      | Procedure | DSQ1BSP                                                          | QMF stored procedure interface |
| Q.DSQABA1E       | DSQABA1E      | Procedure | DSQ1BUDF                                                         | Enhanced list view processing  |
| Q.DSQABA1E       | DSQABA1E_D    | Function  | DSQ1BUDF                                                         | Enhanced list view processing  |
| Q.APPL_AUTHNAMES | DSQABA1E_F0V  | Function  | DSQ1BUDF                                                         | Enhanced list view processing  |
| Q.APPL_AUTHNAMES | DSQABA1E_F1V  | Function  | DSQ1BUDF                                                         | Enhanced list view processing  |

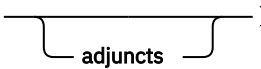
**Important:** If your WLM environment changes after QMF has been installed, you must run ALTER statements to update the WLM environment information for any of the user-defined functions or procedures that you are using. For example:

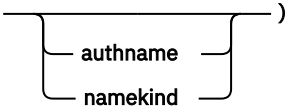
```
ALTER PROCEDURE Q.DSQQMFSP
WLM ENVIRONMENT new_environment_name;
ALTER FUNCTION Q.DSQABA1E
WLM ENVIRONMENT new_environment_name;
ALTER SPECIFIC FUNCTION Q.DSQABA1E_F0V
WLM ENVIRONMENT new_environment_name;
ALTER SPECIFIC FUNCTION Q.DSQABA1E_F1V
WLM ENVIRONMENT new_environment_name;
ALTER PROCEDURE Q.DSQABA1E
WLM ENVIRONMENT new_environment_name;
```

## APPL\_AUTHNAMES

The APPL\_AUTHNAMES function returns the Db2 authorization IDs for the current application process. A row is returned for each authorization name. The schema name is Q.

The syntax description for the user-defined function table is:

➔ APPL\_AUTHNAMES(  ) ➔

➔ RETURNS TABLE(  ) ➔

### adjuncts

VARCHAR(255): A string of authorization names. Specify each authorization name as an identifier or a delimited identifier. Separate each authorization name by one or more blanks. For example:

```
'SALES "DEPT A1" PAYROLL'
```

These three names would be added to the output of the function should they represent distinct values not already defined as authorization IDs for the current process.

The result of the function is a Db2 table with the following columns:

| <i>Table 92. Table created from the APPL_AUTHNAMES function</i> |                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------------------------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Column name                                                     | Data type      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| authname                                                        | CHARACTER(128) | The name for an authorization ID of the current process.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| namekind                                                        | CHARACTER(1)   | A classification code for the name value in AUTHNAME:<br><b>1</b> Primary authorization ID or user name<br><b>2</b> Secondary authorization ID or group name<br><b>3</b> Current authorization ID<br>This applies only when the current SQL ID is neither the primary ID nor a secondary ID of the current process.<br><b>9</b> Adjunct name value<br>This applies only when the ADJUNCT parameter is used and the identifiers it specifies are not authorization IDs of the current process. |

## CALL DSQABA1E

The DSQABA1E stored procedure returns the Db2 authorization IDs for the currently running process. The schema name is Q.

The syntax description for the stored procedure interface is:

➔ CALL — DSQABA1E — ( — userid — , — groupids — , — sqlid — ) ➔



**userid**

VARCHAR(130): The primary authorization ID is returned in the parameter.

**groupids**

VARCHAR(32672): The secondary authorization IDs are returned in this parameter. Each authorization name is converted from a VARCHAR data format and to a single-string structure. The calling program must interpret the content of the character string to obtain the individual authorization names.

**sqlid**

VARCHAR(130): The current SQL authorization ID is returned in this parameter.

## DSQABA1E

---

The DSQABA1E function returns diagnostic information that can assist IBM Software Support with problem diagnosis. The schema name is Q.

The syntax description for the diagnostic user-defined function is:

►► DSQABA1E — ( — ) -►►

The result of the function is a character string with a data type of VARCHAR and an actual length not greater than 5,300 bytes. This string is suitable for formatting in a QMF report with a column width of 53 and an EDIT code of CW.



---

# Appendix F. How QMF and GDDM programs are defined to CICS

QMF for TSO and CICS provides the jobs necessary to define QMF programs to CICS and load GDDM definitions and chart formats for QMF panels.

---

## How QMF programs are defined to CICS

During installation, the default transaction ID QMF $n$  is defined for QMF. The transaction ID is defined in the system definition (CSD) file.

### Resident QMF programs

During installation, the following programs are defined as resident in CICS:

- DSQQMF
- DSQQMF $n$
- DSQCBST
- DSQC $n$ LTT
- DSQC $n$ BLT
- DSQU $n$ GV3
- DSQUECIC

The variable  $n$  in the program names and transaction ID represents the 1-character national language identifier (NLID) that corresponds to the language in which you are running QMF.

CICS treats programs with RMODE(31) as permanently resident because of the large amount of virtual storage available above the 16 MB line. Programs defined as resident are loaded during CICS system initialization. Nonresident programs are loaded on the first reference to the program.

The first QMF transaction to start causes certain GDDM programs to be loaded.

### How nonresident programs affect performance

If several users use QMF, removing QMF programs from resident storage might affect QMF and CICS performance, because QMF must be loaded each time a user starts the transaction. However, if the needs of your site require that you remove these programs from resident storage, change the definition for QMF programs from resident to nonresident.

You can specify RESIDENT=NO on the CEDA DEFINE PROGRAM command to interactively change the program definition in the CSD file.

### Related concepts

[How GDDM definitions are loaded during installation of QMF](#)

QMF uses GDDM services for printing and displaying QMF screens.

### Related tasks

[Installing QMF National Language Features](#)

A QMF National Language Feature (NLF) provides you with an environment that is customized for a specific language. In general, functions in QMF that are available in the base English-language product are also available in NLFs.

## How GDDM definitions are loaded during installation of QMF

---

QMF uses GDDM services for printing and displaying QMF screens.

The VSAM panel file, DSQPNLn, contains text for QMF screens and is described to CICS during installation. QMF also uses the GDDM-PGF product to create charts of many types, such as scatter, pie, histogram, and others.

### How nonresident GDDM programs affect QMF

GDDM programs are not predefined as resident. When you customize GDDM for CICS, consider making the GDDM programs resident, because certain GDDM programs are loaded when QMF is started, whether or not you use the charting functions of QMF.

### How chart formats are defined

The default installation of QMF stores chart formats, chart data, and GDF data in the GDDM ADMF file. You can change the name of this GDDM object file or create additional GDDM object files to store chart objects by modifying the OBJFILE section of the GDDM external defaults module, ADMADFC. For example, you might have separate files for chart formats, chart data, and GDF data.

### Adding charting functions after installation of QMF

If you install GDDM-PGF after you install CICS, you need to fully install and customize GDDM-PGF for CICS, rather than merely restoring the product to a sublibrary. After you install GDDM-PGF and customize it, you can verify the installation by running the CICS ADMC transaction, which is predefined by GDDM during GDDM customizing for CICS. No further customization of the chart formats is necessary; these formats were defined for you during installation of QMF.

### Related concepts

[Installation prerequisites for requester \(Db2 for z/OS\) databases](#)

Before you can install QMF on Db2 for z/OS databases that function as stand-alone or requester databases, you must fulfill the hardware and software requirements.

### Related information

Search for information about customizing GDDM for CICS in the GDDM documentation.

## Transaction routing to control resource use in CICS

---

To protect high-speed transactions in your system from potential long-running QMF queries that might consume extra resources, consider isolating execution of QMF transactions to a single region, using multiregion operations or intersystem communications.

Define one CICS terminal-owning region and route QMF transaction requests to other regions by using multiple transaction IDs or dynamic routing exits.

## Notices

---

This information was developed for products and services offered in the US. This material may 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 give 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 (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 supply 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.

#### 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 shown below.

© (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 publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of QMF.

## Trademarks

---

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)<sup>®</sup> 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 <http://www.ibm.com/legal/copytrade.shtml>.

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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

# Glossary of terms and acronyms

---

**abnormal end of task (abend)**

The termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve during execution.

**address space**

The range of addresses available to a computer program or process. Address space can refer to physical storage, virtual storage, or both.

**Advanced Program-to-Program Communication**

See *APPC*.

**aggregate function**

Any of a group of functions that summarizes data in a column. They are requested with these usage codes on the form panels: AVERAGE, CALC, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, STDEV, SUM, CSUM, PCT, CPCT, TPCT, TCPCT.

**aggregation variable**

An aggregation function that is placed in a report using the FORM.BREAK, FORM.CALC, FORM.DETAIL, or FORM.FINAL panels. Its value appears as part of the break footing, detail block text, or final text when the report is produced.

**alias**

An alternative name used to identify a table, view, database, or nickname. An alias can be used in SQL statements to refer to a table, view, or database in the same Db2 system or subsystem or in a remote Db2 system or subsystem.

**APAR (Authorized Program Analysis Report)**

A request for correction of a defect in a supported release of an program supplied by IBM.

**APF (authorized program facility)**

In a z/OS environment, a facility that permits the identification of programs that are authorized to use restricted functions.

**API (application programming interface)**

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

**application**

One or more computer programs or software components that use QMF services to provide functionality in direct support of a specific business process or processes.

**APPC (Advanced Program-to-Program Communication)**

An implementation of the SNA LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

**application plan**

The control structure that is produced during the bind process. The default name for the QMF Version 12.1 application plan is QMF12.

**application programming interface**

See *API*.

**application requester**

The source of a request to a remote DRDA-enabled relational database management system (RDBMS). Only Db2 for z/OS databases can function as application requesters because this is the only type of database in which QMF can be started.

**application server**

The target of a request from an application requester. The database management system (DBMS) at the application server site services the request. Connectivity with remote servers is not supported when QMF for TSO is running as a Db2 for z/OS stored procedure.

**argument**

A value passed to or returned from a function or procedure at run time.

**authorization identifier (authorization ID)**

A character string that designates a set of privileges and can be used to verify authority. An authorization ID can represent an object, an individual user, an organizational group, a function, or a database role. QMF authenticates either the database authorization ID or, optionally, the QMF TSO logon ID, against the CREATOR column of the Q.PROFILES table during QMF initialization.

**Authorized Program Analysis Report**

See *APAR*.

**Authorized program facility**

See *APF*.

**auxiliary table**

A table that stores columns outside the table in which they are defined. See also *base table*.

**base product**

The English-language version of QMF, established when QMF is installed. Any other language environment is established after installation by installing the National Language Feature (NLF) associated with that language.

**base table**

A table that is created by the SQL CREATE TABLE statement and that holds persistent data.

**binary string**

A sequence of bytes that is not associated with a coded character set and therefore is never converted. For example, the BLOB data type is a binary string. See also *CCSID*.

**bind**

To convert the output from the DBMS precompiler to a usable control structure, such as an access plan, an application plan, or a package.

**bit data**

Data with a data type of CHAR or VARCHAR that is not associated with a coded character set and therefore is never converted.

**buffer pool**

An area of memory into which data pages are read and in which they are modified and held during processing. See also *address space*.

**built-in function**

A strongly typed, high-performance function that is integral to the Db2 database. A built-in function can be referenced in SQL statements anywhere that an expression is valid.

**CAF (call attachment facility)**

A Db2 for z/OS attachment facility for application programs that run in TSO or z/OS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment.

**call attachment facility**

See *CAF*.

**callable interface**

A programming interface that provides access to QMF objects and services.

**cascade delete**

A process by which the Db2 database manager enforces referential constraints by deleting all descendent rows of a deleted parent row.

**catalog**

A collection of tables and views that contains descriptions of objects such as tables, views, and indexes. See also *QMF object catalog*.

**CCSID (coded character set identifier)**

A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character



representation. Because QMF uses display services provided by GDDM, the GDDM application code page must agree with the CCSIDs in use for the database. See also *binary string*.

**character string**

A sequence of bytes that represents bit data, single-byte characters, or a mixture of single-byte and multibyte characters.

**check constraint**

A user-defined constraint that specifies the values that specific columns of a base table can contain. See also *constraint*.

**CICS (Customer Information Control System)**

An IBM licensed program that provides online transaction-processing services and management for business applications.

**clause**

In SQL, a distinct part of a statement in the language structure, such as a SELECT clause or a WHERE clause.

**CM (Compatibility Mode)**

An installation mode of QMF Version 8.1 and QMF Version 9.1 that limited owner and object names in the QMF object catalog to eight and 18 characters, respectively. See also *NFM*.

**code page**

A particular assignment of code points to graphic characters. Within a given code page, a code point can have only one specific meaning. A code page also identifies how undefined code points are handled.

**coded character set identifier**

See *CCSID*.

**coexistence**

The state during which two QMF releases exist in the same Db2 subsystem. QMF Version 12.1 can coexist with QMF Version 9.1 New Function Mode or QMF Version 8.1 New Function Mode only.

**column**

The vertical component of a database table. A column has a name and a particular data type (for example, character, decimal, or integer).

**column function**

See *aggregate function*.

**column wrapping**

The value formatting in a report where the values occupy several lines within a column. Column wrapping is often used when a column contains values whose length exceeds the column width, such as cases requiring the display of XML data.

**command interface**

An interface for issuing QMF commands. The command interface allows you to issue QMF commands from an ISPF dialog running under QMF. Using this interface, QMF communicates with the dialog through the ISPF variable pool.

**command synonym**

The verb or verb/object part of a site-defined command. After command synonyms are defined and activated in the QMF profile, users can enter the synonyms on the QMF command line as they do with regular QMF commands.

**command synonym table**

A table that stores one site-defined command in each row. You assign a set of command synonyms to a user by storing the name of this table in the user's profile.

**comparison operator**

In SQL, a symbol used in comparison expressions to specify a relationship between two values. Comparison operators are = (equal to), <> (not equal to), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to).

**Compatibility Mode**

See *CM*.

**commit**

To end a unit of work by releasing locks so that the database changes made by that unit of work can be perceived by other processes. This operation makes the data changes permanent.

**concatenation**

Joining two characters or strings to form one string.

**connection**

In data communication, an association established between entities for conveying information. See also *SQL connection*. Connectivity with remote servers is not supported when QMF for TSO is running as a Db2 for z/OS stored procedure.

**constant**

A language element that specifies an unchanging value. Constants are classified as string constants or numeric constants.

**constraint**

A rule that limits the values that can be inserted, deleted, or updated in a table.

**control section**

See *CSECT*.

**control tables**

A set of tables that QMF uses to store information about QMF objects and manage QMF operations. See also *QMF object catalog*.

**correlated reference**

A reference to a column of a table or view that is outside a subquery.

**correlation name**

An identifier specified and used within a single SQL statement as the exposed name for objects such as a table, view, table function reference, nested table expression, or data change table reference. Correlation names are useful in an SQL statement to allow two distinct references to the same base table and to allow an alternative name to be used to represent an object.

**CSECT (control section)**

The part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining main storage locations.

**current location**

The application server to which the QMF session is currently connected. After the connection is made, this server processes all SQL statements. When initializing QMF, the current location can be indicated using the DSQSDBNM startup parameter. Connectivity with remote servers is not supported when QMF for TSO is running as a Db2 for z/OS stored procedure.

**current object**

A QMF object that is held in temporary storage so that, with each use, it can be readily accessed without requiring database retrieval. There are seven temporary storage areas: QUERY, FORM, PROC, PROFILE, REPORT, DATA, and CHART. Users can navigate to all areas but the DATA area using the SHOW and DISPLAY commands. See also *temporary storage*.

**cursor**

A named control structure used by an application program to point to and select a row of data from a set.

**Customer Information Control System**

See *CICS*.

**data type**

A classification identifying one of various kinds of data. In SQL, the data type is an attribute of columns, literals, host variables, special registers, parameters, and the results of functions and expressions.

**database**

A collection of interrelated or independent data items that are stored together to serve one or more applications.

**database administrator**

A person who is responsible for the design, development, operation, security, maintenance, and use of a database.

**database management system**

See *DBMS*.

**database manager**

A program that manages data by providing centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency control, privacy, and security.

**database server**

A software program that uses a database manager to provide database services to other software programs or computers.

**DBCS (double-byte character set)**

A set of characters in which each character is represented by two bytes. These character sets are commonly used by national languages such as Japanese and Chinese, which have more symbols than can be represented by a single byte. See also *SBCS*.

**DBMS (database management system)**

A software system that controls the creation, organization, and modification of a database and the access to the data that is stored within it.

**DCT (destination control table)**

A table describing each of the transient data destinations used in CICS. This table contains an entry for each extrapartition, intrapartition, and indirect destination.

**default form**

The QMF form created when a saved form is not specified on the RUN QUERY command.

**default value**

A predetermined value, attribute, or option that is assumed when no other value is specified. A default value can be defined for column data in Db2 tables by specifying the DEFAULT keyword in an SQL statement that changes data (such as INSERT, UPDATE, and MERGE).

**dependent row**

A row that contains a foreign key that matches the value of a parent key in the parent row. The foreign key value represents a reference from the dependent row to the parent row.

**dependent table**

A table that is a dependent of an object. For example, a table with a foreign key is a dependent of the table containing the corresponding primary key.

**destination control table**

See *DCT*.

**detail block text**

The text in the body of a report that is associated with a particular row of data.

**detail heading text**

The text in the heading of a report.

**detail variation**

A data formatting definition specified on a FORM.DETAIL panel that can be used to conditionally format a report or part of a report.

**distinct type**

A user-defined data type that shares a common representation with built-in data types.

**distributed data**

Data that is stored on more than one system and is available to remote users and application programs.

**distributed database**

A database that appears to users as a logical whole, locally accessible database, but consists of databases in multiple locations that are connected by a data communications network.

**Distributed Relational Database Architecture™**

See *DRDA*.

**distributed unit of work**

A form of distributed relational database processing that enables a user or application program to read or update data at multiple locations within a unit of work. Within one unit of work, an application, such as QMF, running in one system can direct SQL requests to multiple remote database management systems using the SQL supported by those systems. The request is made through a QMF command that includes a three-part table or view name. QMF commands with three-part names cannot be directed to Db2 for VM or VSE databases or used when QMF for TSO has been started as a Db2 for z/OS stored procedure. Three-part names in QMF commands also cannot refer to a table that contains large object (LOB) data types.

**double-byte character set**

See *DBCS*.

**double-precision floating-point number**

A 64-bit approximate representation of a real number.

**DRDA (Distributed Relational Database Architecture)**

The architecture that defines formats and protocols for providing transparent access to remote data. DRDA defines two types of functions: the application requester function and the application server function.

**environment**

A named collection of logical and physical resources used to support the performance of a function.

**exit routine**

A program that receives control from another program to perform specific functions.

**Extensible Markup Language**

See *XML*.

**extended syntax**

Syntax that is used for the QMF SET GLOBAL and GET GLOBAL commands and certain function calls in a callable interface application. Extended syntax defines parameters used by QMF callable interface applications written in Assembler, C, COBOL, Fortran, or PL/I.

**fallback**

The process of returning to a prior release of a software program after attempting or completing migration to a current release.

**fetch**

The process of retrieving rows from the database or a file to create a QMF DATA object. QMF supports multirow fetch through the use of the DSQSMRFI parameter.

**foreign key**

In a relational database, a key in one table that references the primary key in another table.

**GDDM (Graphical Data Display Manager)**

Graphics software that defines and displays text and graphics for output on a display device or printer.

**global variable**

A named entity whose value persists for the duration of a QMF session by default. QMF uses global variables to manage both session and database activity. Some global variables can be set with the SET GLOBAL command, while others record information about the state of the current QMF session and therefore cannot be set.

**graphic string**

A sequence of double-byte character set (DBCS) characters.

**Graphical Data Display Manager**

See *GDDM*.

**host**

The controlling or highest-level system in a data communications configuration.

**HTML (hypertext markup language)**

A markup language that conforms to the Standard Generalized Markup Language (SGML) standard and was designed primarily to support the online display of textual and graphical information, including hypertext links.

**hypertext markup language**

See *HTML*.

**ICU (Interactive Chart Utility)**

A menu-driven component of IBM's Graphical Data Display Manager (GDDM) product that allows non-programmers to display, print, or plot charts, graphs, and diagrams.

**identity column**

A column that provides a way for the Db2 database manager to automatically generate a numeric value for each row that is inserted into a table. Identity columns are defined with the AS IDENTITY clause. A table can have no more than one identity column.

**index**

A set of pointers that is logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness of the key values for the rows in the table.

**inner join**

The result of a join operation that includes only the matched rows of both tables that are being joined. See also *outer join*.

**installation verification procedure**

See *IVP*.

**Integrated Exchange Format**

See *IXF*.

**Interactive Chart Utility**

See *ICU*.

**Interactive System Productivity Facility**

See *ISPF*.

**ISPF (Interactive System Productivity Facility)**

An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and terminal user.

**IVP (installation verification procedure)**

A procedure or program whose purpose is to verify that a product has been correctly installed.

**IXF (Integrated Exchange Format)**

A protocol for transferring tabular data among various software products.

**JCL (job control language)**

A command language that identifies a job to an operating system and describes the job's requirements.

**job control language**

See *JCL*.

**join**

An SQL relational operation that allows retrieval of data from two or more tables based on matching column values.

**key**

A column or an ordered collection of columns that is identified in the description of a table, index, or referential constraint. The same column can be part of more than one key.

**keyword**

One of the predefined words of a programming language, artificial language, application, or command.

**keyword parameter**

A parameter that consists of a keyword followed by one or more values. See also *positional parameter*.

**large object**

See *LOB*.

**link-edit**

To create a loadable computer program by means of a linkage editor.

**linkage editor**

A computer program for creating load modules from one or more object modules or load modules by resolving cross-references among the modules and, if necessary, adjusting addresses.

**literal**

A character string whose value is defined by the characters themselves. For example, the numeric constant 7 has the value 7, and the character constant 'CHARACTERS' has the value CHARACTERS.

**linear procedure**

A sequenced set of QMF commands or command synonyms that can be used to perform several operations at once. See also *procedure with logic*.

**linear syntax**

QMF command syntax that is entered in one statement of a program or procedure, or that can be entered on the QMF command line.

**load module**

A program in a form suitable for loading into main storage for execution.

**LOB (large object)**

A sequence of bytes with a size ranging from 0 bytes to 2 gigabytes (less 1 byte). There are three LOB data types: binary large object (BLOB), character large object (CLOB, which can include single-byte characters only or a mixture of single-byte and double-byte characters), and double-byte character large object (DBCLOB). QMF supports a LOB column size of up to 32 KB.

**local**

Pertaining to databases, objects, or applications that are installed or stored in the system in which QMF is currently running.

**location**

A specific relational database server in a distributed relational database system. Each location has a unique location name.

**location name**

The unique name of a database server. An application uses the location name to access a Db2 database server.

**lock**

A means of serializing a sequence of events or serializing access to data.

**log**

A collection of records that sequentially describes the events that occur in a system.

**LUW**

An abbreviation for Linux, UNIX, and Windows.

**National Language Feature**

See *NLF*.

**New Function Mode**

See *NFM*.

**NFM (New Function Mode)**

An installation mode of QMF Version 8.1 and QMF Version 9.1 that allowed owner and object names in the QMF object catalog to be the maximum length allowed by the database. QMF Version 12.1 allows owner and object names to be as long as the database allows as well. See also *CM*.

**NLF (National Language Feature)**

Any of several optional features available with QMF. NLFs allow users to interact with QMF in specific native languages.

**object**

A named storage space that consists of a set of characteristics that describe the space and, in some cases, data. An object is anything that occupies space in storage, can be located in a library or directory, can be secured, and on which defined operations can be performed. See also *QMF object*.

**outer join**

The result of a join operation that includes the matched rows of both tables that are being joined and preserves some or all of the unmatched rows of the tables that are being joined. See also *inner join*.

**package**

A control-structure database object produced during program preparation that can contain both executable forms of static SQL statements or XQuery expressions and placement holders for executable forms of dynamic SQL statements.

**panel**

A formatted display of information on a screen that can also include entry fields.

**parameter**

A value or reference passed to a function, command, or program that serves as input or controls actions. The value is supplied by a user or by another program or process.

**partition**

A portion of a page set. Each partition corresponds to a single, independently extendable data set. Partitions can be extended to a maximum size of 1, 2, or 4 gigabytes, depending on the number of partitions in the partitioned page set. All partitions of a given page set have the same maximum size.

**plan**

See *application plan*.

**positional parameter**

A parameter that must appear in a specified location, relative to other parameters. See also *keyword parameter*.

**precision**

An attribute of a number that describes the total number of significant digits.

**predicate**

An element of a search condition that expresses or implies a comparison operation.

**primary authorization ID**

The authorization identifier used to identify an application process to Db2 for z/OS.

**primary key**

In a relational database, a key that uniquely identifies one row of a database table.

**privilege**

In SQL, a capability given to a user by the processing of a GRANT statement.

**procedure**

A sequenced set of statements or commands used to perform one or more tasks. See also *linear procedure* and *procedure with logic*.

**procedure with logic**

A set of statements that performs one or more tasks. A procedure with logic begins with a REXX comment and allows conditional logic (which uses REXX), calculations, build strings, and TSO or CICS commands. See also *linear procedure*.

**profile**

An object that contains information about the characteristics of the user's session.

**program temporary fix**

See *PTF*.

**prompted query**

A menu-driven query controlled by user-provided parameters.

**PTF (program temporary fix)**

For System i®, System p, and System z®, products, a fix that is tested by IBM and is made available to all customers.

**QBE (Query-by-Example)**

A component of QMF that allows users to create queries graphically.

**QMF administrator authority**

Authority that allows a user to insert or delete rows in the Q.PROFILES control table. Users with this authority can perform the following commands on QMF queries, forms, and procedures that are owned by other users without forcing the owners to share these objects with all users: SAVE, ERASE, IMPORT, EXPORT, and DISPLAY. QMF checks each user ID for administrator authority during

initialization; you can disable this checking by setting the DSQEC\_DISABLEADM variable in the DSQUOPTS exit routine or in another program of your choice.

**QMF administrator**

A user who has QMF administrator authority.

**Query-by-Example**

See *QBE*.

**QMF object**

An object used by QMF users to query, format, and present data or otherwise manage interaction between QMF and the database. QMF objects include queries and query result data, forms, procedures, reports, charts, and the QMF profile. Each QMF object has a named temporary storage area that is used to display the object. All objects except reports and charts can be saved in the database; reports and charts are created dynamically upon user request by applying the formatting specifications of a particular QMF form to result data that has been returned from the database. See also *temporary storage*.

**QMF object catalog**

A set of control tables that stores information about QMF queries, procedures, forms, folders, and analytics objects. These control tables include Q.OBJECT\_DIRECTORY, Q.OBJECT\_DATA, and Q.OBJECT\_REMARKS.

**qualifier**

When referring to a QMF object, the part of the name that identifies the owner or the location of an object. When referring to a TSO data set, any part of the name that is separated from the rest of the name by periods. For example, 'TCK', 'XYZ', and 'QUERY' are all qualifiers in the data set name 'TCK.XYZ.QUERY'.

**query**

A request for information from a database based on specific conditions: for example, a request for a list of all customers in a customer table whose balances are greater than \$1000. In QMF, a query also refers to SQL statements submitted from the Prompted Query, QBE, or SQL query panel, even if these statements are not requests for information (SELECT statements).

**RCT (resource control table)**

A Db2 control table that defines the relationship between CICS transactions and Db2 resources.

**RDBMS (relational database management system)**

A collection of hardware and software that organizes and provides access to a relational database.

**RDO (resource definition online)**

In CICS, a facility that allows the user to define certain CICS resources interactively while CICS is running. Specifically, RDO allows the user to define terminals, programs, and transactions interactively.

**record**

The storage representation of a row or other data.

**record length**

The length of storage that represents a row or other data.

**reentrant**

Executable code that can reside in storage as one shared copy for all database threads. Reentrant code is not self-modifying and provides separate storage areas for each thread.

**referential constraint**

The requirement that the nonnull values of a designated foreign key are valid only if they also appear as values of the primary key of the parent table. The referential constraint is always defined from the perspective of the dependent file.

**relational database**

A database that can be perceived as a set of tables and manipulated in accordance with the relational model of data. Each database includes a set of system catalog tables that describe the logical and physical structure of the data, a configuration file containing the parameter values allocated for the database, and a recovery log with ongoing transactions and archivable transactions.



**relational database management system**

See *RDBMS*.

**remote**

Pertaining to databases, objects, or applications that are installed or stored on a system other than the system where QMF is currently executing. You can access objects (including QMF queries, forms, procedures, folders, and analytics objects) at a remote server by using the QMF CONNECT command. You can also use a QMF command with a three-part table or view name if you want to access just tables or views at a remote location. Remote access is not permitted when QMF for TSO is running as a Db2 for z/OS stored procedure.

**remote unit of work**

A form of distributed relational database processing in which an application program, such as QMF, can access data on a remote database within a unit of work. The connection is established by the QMF CONNECT command. The CONNECT command cannot be used when QMF for TSO is running as a Db2 for z/OS stored procedure.

**requester**

See *application requester*.

**resource**

The object of a lock or claim, which could be a table space, an index space, a data partition, an index partition, or a logical partition.

**resource control table**

See *RCT*.

**resource definition online**

See *RDO*.

**Restructured Extended Executor**

See *REXX*.

**REXX (Restructured Extended Executor)**

A general-purpose, high-level programming language, particularly suitable for EXEC procedures or programs for personal computing.

**roll back**

To restore data that is changed by an SQL statement to the state at its last commit point. If a failure occurs in a query that contains multiple statements and no COMMIT statements, all statements, except those that affect the QMF session (such as SET), are rolled back. If a failure occurs in a query that contains one or more COMMIT statements, all updates after the last successful COMMIT statement are rolled back. In either case, the query ends after the failure.

**routine**

A program or sequence of instructions called by a program. Typically, a routine has a general purpose and is frequently used.

**row**

The horizontal component of a table, consisting of a sequence of values, one for each column of the table.

**runtime variable**

A variable in a procedure or query whose value is specified by the user when the procedure or query is run. The value of a runtime variable is only available in the current procedure or query. See also *global variable*.

**SBCS (single-byte character set)**

A coded character set in which each character is represented by a 1-byte code. A 1-byte code point allows representation of up to 256 characters. See also *double-byte character set*.

**scalar function**

An SQL function that optionally accepts arguments and that returns a single scalar value each time that it is invoked. A scalar function can be referenced in an SQL statement wherever an expression is valid.

**scratchpad area**

A work area used in conversational processing to retain information from an application program across executions of the program.

**search condition**

A criterion for selecting rows from a table. A search condition consists of one or more predicates.

**secondary authorization ID**

In Db2 for z/OS, an authorization identifier that is associated with a primary authorization ID by an authorization exit routine. See also *primary authorization ID*.

**segmented table space**

A table space that is divided into equal-sized groups of pages called segments. Segments are assigned to tables so that rows of different tables are never stored in the same segment. See also *table space*.

**server**

See *application server*.

**session**

All interactions between the user and QMF from the time the user invokes QMF until the EXIT command is issued.

**shift-in character**

A control character (X'0F') that is used in EBCDIC systems to denote that the subsequent bytes represent SBCS characters. See also *shift-out character*.

**shift-out character**

A control character (X'0E') that is used in EBCDIC systems to denote that the subsequent bytes, up to the next shift-in control character, represent DBCS characters. See also *shift-in character*.

**single-byte character set**

See *SBCS*.

**single-precision floating-point number**

A 32-bit approximate representation of a real number.

**SQL (Structured Query Language)**

A standardized language for defining and manipulating data in a relational database.

**SQL authorization ID**

See *SQLID*.

**SQL connection**

An association between an application process and a local or remote application server or database server. See also *remote unit of work*, *distributed unit of work*.

**SQL function**

A function that is implemented entirely by using a subset of SQL statements and SQL PL statements.

**SQL ID (SQL authorization ID)**

In Db2 for z/OS, the ID that is used for checking the authorization of dynamic SQL statements in some situations.

**SQL return code**

The SQLSTATE or SQLCODE that indicates whether the previously run SQL statement completed successfully, with one or more warnings, or with an error.

**SQLCA (Structured Query Language Communication Area)**

A set of variables that provides an application program with information about the execution of its SQL statements or requests from the database manager. When an error is associated with an SQL code, the QMF message help (available by pressing the Help key) displays the contents of the SQLCA.

**stored procedure**

A routine that can be invoked using the SQL CALL statement to perform operations that can include both host language statements and SQL statements.

**stored procedure interface**

An interface to QMF for TSO that allows you to start QMF as a Db2 for z/OS stored procedure, pass the name of a QMF query or procedure that performs the work you require, and receive up to 21 result

sets back, including a result set for trace output. QMF for TSO can be started in this manner from any product that can run a Db2 for z/OS stored procedure.

**Structured Query Language**

See *SQL*.

**Structured Query Language Communication Area**

See *SQLCA*.

**subquery**

A complete SQL query that appears in a WHERE or HAVING clause of another query.

**substitution variable**

(1) A variable in a procedure or query whose value is specified either by a global variable or by a runtime variable. (2) A variable in a QMF form whose value is specified by a global variable.

**substring**

A part of a character string.

**subsystem**

In Db2 for z/OS, a distinct instance of a relational database management system (RDBMS).

**table**

In a relational database, a database object that consists of a specific number of columns and is used to store an unordered set of rows. See also *base table*.

**table space**

A logical unit of storage in a database. In Db2 for z/OS, a table space is a page set and can contain one or more tables. In Db2 for Linux, UNIX, and Windows, a table space is a collection of containers, and the data, index, long field, and LOB portions of a table can be stored in the same table space or in separate table spaces.

**temporary storage**

An area used to store a QMF object temporarily while the user is working on it so that, with each use, it can be readily accessed without further database retrieval. There are seven temporary storage areas: QUERY, DATA, FORM, PROC, REPORT, CHART, or PROFILE. With the exception of query result data (the DATA object), the QMF objects in these areas can be displayed using the SHOW command followed by the name of the storage area. Though the contents of the DATA area cannot be directly displayed, users can issue the SHOW REPORT or SHOW CHART commands to see the query result data formatted with the specifications of the form currently in the FORM area. See also *QMF object*, *current object*.

**temporary storage queue**

In CICS, a queue of data items which can be read and reread, in any sequence. The queue is created by a task, and persists until the same task or a another task deletes it. See also *transient data queue*.

**thread**

The Db2 structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to Db2 resources and services. Most Db2 functions execute under a thread structure.

**three-part name**

The full name of a table, view, or alias that consists of a location name, an authorization identifier, and an object name, separated by periods. QMF commands that include three-part names can be initiated only from Db2 for z/OS databases and can be directed to all databases except DB2 for VM or VSE.

When QMF for TSO has been started as a Db2 for z/OS stored procedure, QMF commands with three-part names are not supported.

**Time Sharing Option**

See *TSO*.

**trace**

A record of the processing of a computer program or transaction. The information collected from a trace can be used to assess problems and performance.

**transaction**

A unit of processing consisting of one or more application programs, affecting one or more objects, that is initiated by a single request.

**transient data queue**

A CICS storage area where objects are stored for subsequent internal or external processing. See also *temporary storage queue*.

**trigger**

A database object that is associated with a single base table or view and that defines a rule. The rule consists of a set of SQL statements that runs when an insert, update, or delete database operation occurs on the associated base table or view.

**TSO (Time Sharing Option)**

A base element of the z/OS operating system that allows users to work interactively with the system.

**two-phase commit**

A two-step process by which recoverable resources in an external subsystem are committed. During the first step, the database manager subsystems are polled to ensure that they are ready to commit. If all subsystems respond positively, the database manager instructs them to commit.

**UDF (user-defined function)**

A function that is defined to the Db2 database system by using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A UDF can be an external function or an SQL function.

**Unicode**

A character encoding standard that supports the interchange, processing, and display of text that is written in the common languages around the world, plus some classical and historical texts. The Unicode standard has a 16-bit character set defined by ISO 10646.

**unit of recovery (UR)**

A sequence of operations within a unit of work between points of consistency.

**unit of work (UOW)**

A recoverable sequence of operations within an application process. At any time, an application process is a single UOW, but the life of an application process can involve many UOWs as a result of commit or rollback operations. In a multisite update operation, a single UOW can include several units of recovery. In QMF SQL queries that include multiple statements and no COMMIT statements, all statements comprise a single unit of work, so all statements except those that affect the session (such as SET) are rolled back in the event of a failure. In QMF SQL queries that include multiple statements and one or more COMMIT statements, a unit of work consists of a COMMIT statement and all previous statements back to the beginning of the query or the last COMMIT statement. If a failure occurs, all updates after the last successful COMMIT statement are rolled back.

**user-defined function**

See *UDF*.

**view**

A logical table that is based on data stored in an underlying set of tables. The data returned by a view is determined by a SELECT statement that is run on the underlying tables.

**XML (Extensible Markup Language)**

A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML).

**z/OS**

An IBM mainframe operating system that uses 64-bit real storage.

# Index

## Numerics

64-bit storage for report operations (TSO) [152](#)

## A

abbreviating command synonyms [236](#)

ABEND, from QMF install [338](#)

ABENDASRA [338](#)

abends

DCB intercepts, spill file considerations [349](#)

access

to QMF application plan packages [171](#)

ADD mode, Table Editor

governing [315](#)

ADMADFC defaults module [218](#)

ADMCFORM ddname [225](#)

administration

authority required [13](#)

tables, creating [197](#)

user profiles and objects [184](#)

administrator authority, *See* QMF administrator authority

ADMMNICK specification [214](#)

AEY9 abend [338](#)

ALLOBJ authority prerequisites (iSeries) [13](#)

allocation of data sets

stored procedure interface [107](#)

allocation of storage

dynamic (TSO) [151](#)

fixed (TSO, CICS) [151](#)

percentage (TSO) [151](#)

spill data

extended storage (TSO) [152](#)

file (CICS) [157](#)

file (TSO) [153](#)

overview [152](#)

variable (TSO) [151](#)

ampersand (&)

in command synonyms [232](#)

analytic objects

displaying [208](#)

analytics objects

internal stored format [204](#)

listing [207](#)

APAR (Authorized Program Analysis Report) [335](#), [363](#), [365](#)

APPCPG parameter, GDDM [49](#), [57](#)

APPLDATA column [204](#)

application code page

CCSID incompatibilities with GDDM [49](#), [57](#)

application plan for QMF

access type [171](#)

required locations [7](#)

ARCH(7) compiler option, user edit routines [268](#), [270](#)

ASCPUT services, printing [214](#)

asynchronous processing, printing [213](#)

asynchronous timer routine [313](#)

authentication

authentication (*continued*)

initial procedure [128](#), [163](#)

national language features [101](#)

primary authorization ID vs. TSO ID [150](#)

QMF administrator authority verification, *See* QMF administrator authority

QMF plan and packages, controlling access [171](#)

QMF profile

adding [181](#)

CREATOR column of Q.PROFILES [172](#)

open enrollment [184](#)

restricted enrollment [180](#)

updating [183](#)

secondary authorization IDs

installation using [13](#)

LIST command support [109](#)

stored procedure interface (TSO only) [107](#)

system initialization procedure (Q.SYSTEM\_INI) [137](#)

authority required for QMF tasks

accessing sample tables [101](#)

administration [13](#)

installation [13](#)

installation verification procedure (IVP) [71](#)

QMF administrator, *See* QMF administrator authority

authority, Db2

distributing, overview [209](#)

authorization

command synonyms [238](#)

error [340](#)

QMF administrator authority, *See* QMF administrator authority

to access QMF [180](#)

authorization ID

authentication during initialization [107](#)

authentication through TSO logon ID [150](#)

checking for QMF administrator authority [131](#)

enrolling individual users versus groups [171](#)

installing under secondary [13](#), [30](#), [40](#), [45](#), [47](#), [62](#), [64](#), [66](#), [109](#)

position in qualified names [8](#)

Q [13](#)

Q.SYSTEM\_INI requirements [137](#)

RACF group names, *See* secondary authorization IDs

requirements for QMF IDs [172](#)

returning list for current application [401](#)

running an initial procedure [128](#)

secondary, *See* secondary authorization IDs

supported lengths [4](#), [6](#)

*See also* authority required for QMF tasks

automatic routing, print output [213](#)

automation of QMF tasks

batch mode QMF

CICS [333](#)

program parameters [162](#)

TSO, ISPF, native z/OS [319](#)

procedures

initialization exit that sets global variables [131](#)

automation of QMF tasks (*continued*)  
  procedures (*continued*)  
  post-initialization [163](#)  
  running in stored procedure interface [107](#), [119](#)  
AZTS abend [338](#)

## B

B37 abends and spill file considerations [349](#)  
backups  
  creating  
    QMF control table listing [387](#)  
  recovery to  
    servers [67](#)  
BACKWARD command  
  governing [315](#)  
backward compatibility between QMF releases [85](#)  
base QMF commands as synonyms [230](#)  
batch  
  running a query or procedure in [227](#)  
  running from remote Db2 client [332](#)  
  using a spill file [157](#)  
BIGINT data type  
  SQLTYPE, user edit routines [254](#)  
  width [153](#)  
BINARY data type  
  SQLTYPE, user edit routines [254](#)  
  width [153](#)  
bit edit codes [347](#)  
BLOB data, user edit routines [253](#)  
BOTTOM command  
  governing [315](#)  
branch addresses, governor [297](#)  
buffer pools, control table assignments for [211](#)

## C

C trace code [353](#)  
calculate spill file size [153](#)  
CALL statement  
  restrictions for multistatement queries [171](#)  
  syntax, stored procedure interface [107](#)  
callable interface  
  common error [340](#)  
cancellation service, governor [313](#)  
carriage control characters  
  eliminating from result sets [119](#)  
cascading authority [188](#)  
CASE column (Q.PROFILES) [172](#)  
catalog  
  QMF objects, *See* object catalog  
catalog views  
  required locations [7](#)  
CCSID incompatibilities [49](#), [57](#)  
CEBR transaction [333](#)  
CHANGE mode, Table Editor  
  governing [315](#)  
CHAR data type  
  SQLTYPE, user edit routines [254](#)  
  width [153](#)  
chart  
  formats [224](#)  
  printing

chart (*continued*)  
  printing (*continued*)  
    specific objects [223](#)  
charting functions, tracing [353](#)  
CICS environment  
  common errors [338](#)  
  EDSA limit size for large QMF queries [60](#)  
  ENVIRONMENT values, QMF profile [172](#)  
  HANDLE CONDITION [294](#)  
  IMPORT command [185](#)  
  interface to governor [290](#)  
  NLF control tables [101](#)  
  NLF customization [101](#)  
  storage requirements  
    initialization [37](#)  
    report operations [37](#)  
    setting fixed storage [151](#)  
  temporary storage queue [221](#)  
  transaction ID [340](#)  
  transient data queue [221](#)  
  TYPETERM entries, QMF display [364](#)  
  unsupported functions [384](#)  
  using the trace facility [356](#)  
CICSenvironment  
  diagnostic facilities [364](#)  
class ID, customizing function keys [246](#)  
cleanup after installation [89](#)  
CLOB data, user edit routines [253](#)  
code page incompatibilities between GDDM and database  
[49](#), [57](#)  
coded character set identifier (CCSID) incompatibilities [49](#),  
[57](#)  
codes, SQL, *See* SQL codes  
coexistence  
  different installation modes [4](#)  
column  
  number supported in queries [383](#)  
command synonym table  
  creating [234](#)  
  maintaining [237](#)  
  views [237](#)  
commands  
  governing through resource limit facility [315](#)  
  RUN  
    synonym definition [231](#)  
  SET PROFILE [183](#)  
  tracing [119](#)  
comments on function keys table [245](#)  
comments, sending to IBM [xiii](#)  
common services in QMF, tracing [353](#)  
communications, establishing between QMF installations,  
*See* connectivity, establishing between QMF installations  
compatibility between QMF releases [4](#), [85](#)  
Compatibility Mode  
  checking for [14](#), [19](#)  
  QMF  
    coexistence with New Function Mode [4](#)  
    prior releases [4](#)  
Compatibility with earlier releases [87](#)  
configuration of QMF  
  allocating files for QMF Analytics for TSO [97](#)  
  customizing QMF Analytics for TSO [98](#), [99](#)  
  customizing z/OS environment preferences [49](#)  
  distributed data access overview [5](#)

- configuration of QMF (*continued*)
  - distributed unit of work [8](#)
  - HFS customization [98](#)
  - installation roadmaps [11](#)
  - installation verification procedure (IVP) [99](#)
  - Load library [97](#)
  - National Language Features (NLFs) [101](#)
  - optional features [97](#), [99](#)
  - planning. *See* [planning for installation](#)
  - QMF Analytics for TSO [97](#)
  - QMF Analytics for TSO file allocation [97](#)
  - QMF Analytics for TSO sample tables [99](#)
  - remote unit of work [5](#), [8](#)
  - SAVE DATA command setup [13](#)
  - server databases [61](#)
  - zFS customization [98](#)
- configuration of QMF Analytics for TSO
  - customizing HFS [98](#)
  - customizing ZFS [98](#)
  - setting up work file location [98](#)
- CONFIRM column (Q.PROFILES) [172](#)
- CONNECT command
  - distributed data access overview [5](#)
  - errors [340](#), [344](#)
  - installation roadmaps [7](#)
  - required configuration [5](#)
  - restrictions [6](#), [8](#)
- connectivity, establishing between QMF installations
  - CONNECT command. *See* [CONNECT command](#)
  - distributed unit of work. *See* [distributed unit of work](#)
  - DRDA requirements [5](#)
  - remote unit of work. *See* [remote unit of work](#)
  - restrictions [6](#), [8](#)
  - three-part names. *See* [three-part names](#)
- contention for resources
  - concurrent access resolution options [351](#)
  - uncommitted read vs. cursor stability [189](#)
- control blocks for QMF exit routines
  - governor
    - DXEGOVA [299](#)
    - DXEXCBA [303](#)
- control tables
  - coexistence of QMF releases [4](#)
  - compatibility of Version 12.1 with prior [4](#)
  - DSQDBCTL database [13](#)
  - maintenance
    - environment [210](#)
  - Q.ERROR\_LOG [362](#)
  - Q.RESOURCE\_VIEW [284](#)
  - required locations [7](#)
  - switching buffer pools [211](#)
  - VSAM clusters for user managed data sets [210](#)
  - when to reorganize [210](#)
- controlling access to QMF [171](#)
- conventions for highlighting [xii](#)
- CONVERT command
  - date last used, setting behavior [201](#)
- CREATE PROCEDURE statement
  - restrictions for multistatement queries [171](#)
- CREATE TABLE statement
  - command synonyms [234](#)
  - privileges for SAVE DATA [185](#)
  - tables for users [197](#)
- CREATETS/CREATETAB privilege
  - privilege to run CREATE TABLE statement [199](#)
- CREATOR column (Q.PROFILES)
  - defined [172](#)
  - role in profile initialization [181](#)
- CSD file
  - customizing for the panel file [338](#)
- CURRENT APPLICATION ENCODING SCHEME special register [49](#), [57](#)
- cursor stability options [189](#)
- customizing QMF Analytics for TSO
  - installation verification procedure (IVP) [99](#)

## D

- D trace code [353](#)
- DASD requirements for QMF [33](#)
- DATA object
  - SAVE DATA configuration [13](#), [185](#)
  - storage. *See* [allocation of storage](#)
- data sets
  - allocation
    - stored procedure interface (TSO) [107](#)
  - management of
    - overview [210](#)
- data type
  - edit exit conversions [258](#)
- database
  - functions that vary from one to another [383](#)
  - uncommitted read vs. cursor stability [189](#)
- database services functions, tracing [353](#)
- databases
  - authority prerequisites [13](#), [209](#)
  - CCSID incompatibilities with GDDM [49](#), [57](#)
  - connection
    - authority [171](#)
    - initial, remote unit of work [6](#)
    - remote [117](#)
  - DSQDBCTL [13](#)
  - fetch program parameters [159](#), [160](#)
  - installation prerequisites [33](#)
  - partition groups
    - DSQTSCTL [13](#)
    - DSQTSOBJ [13](#)
  - prerequisite releases for QMF installation [3](#)
  - slow performance [347](#)
- DATE data type
  - edit routines that format [261](#)
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- date last used, setting behavior [132](#), [201](#)
- Db2
  - governor [281](#)
- Db2 governor
  - resource limit specification table [317](#)
- DBA authority prerequisites (VM/VSE) [13](#)
- DBADM authority prerequisites [13](#), [209](#)
- DBCLOB data, user edit routines [253](#)
- DBCS (double-byte character set) support
  - CCSIDs [49](#), [57](#)
  - device requirements [33](#)
  - edit codes [259](#)
  - Latin characters [259](#)
  - stored procedure interface (TSO) [107](#)

DCB abends and spill file considerations [349](#)

DECFLOAT data type
 

- compiler options for user exit routines
  - PL/I (CEEPIPI LE interface) [268](#)
  - PL/I for CICS [270](#)
- conversion in user edit routines [258](#)
- requirements for support [33](#), [253](#)
- SQLTYPE, user edit routines [254](#)
- width [153](#)

DECIMAL data type
 

- SQLTYPE, user edit routines [254](#)
- width [153](#)

decimal data, edit routine [258](#)

decimal floating-point data type, *See* DECFLOAT data type

DECOPT column (Q.PROFILES) [172](#)

device requirements for QMF [33](#)

DEVTOK keyword, ADMMNICK specification [217](#)

diagnostics
 

- aids [353](#)
- dumps [364](#)
- symptoms [353](#)
- trace facility [356](#)

dialog command processing functions, tracing [353](#)

DISPLAY command
 

- date last used, setting behavior [201](#)
- QMF administrator privileges [13](#)
- SQL privileges required [185](#)

display devices
 

- GDDM nicknames [214](#)
- support for [33](#)

display services, tracing [353](#)

DISPLAY TABLE command
 

- governing [315](#)
- storage, *See* allocation of storage

distributed data access
 

- overview [5](#)

Distributed Relational Database Access, *See* DRDA (Distributed Relational Database Access)

distributed unit of work
 

- databases that do not support [383](#)
- DRDA requirements [5](#)
- installation procedure [68](#)
- installation roadmaps [9](#), [27](#)
- required configuration [8](#)
- restrictions [8](#)

double-byte character set (DBCS) support
 

- edit codes [259](#)
- Latin characters [259](#)

DPRE (display printed report) synonym
 

- customizing [229](#)

DRAW command, SQL privileges required [185](#)

DRDA (Distributed Relational Database Access)
 

- requirements for remote data access [5](#)

DSQ, names that begin with (*continued*)
 

- data sets/libraries
  - DSQDEBUG (QMF trace output) [50](#)
  - DSQEDIT (edit transfer file) [50](#)
  - DSQPLBE (ISPF panel library) [34](#)
  - DSQPNLE (QMF panel library) [34](#), [37](#), [50](#), [75](#)
  - DSQPRINT (QMF print output) [50](#)
  - DSQSPILL (spill file) [50](#)
  - DSQUCFRM (user-defined chart formats) [50](#)
  - DSQUDUMP (QMF snap dump) [50](#)
- database partition groups (*continued*)
  - DSQTSTCTL (QMF control tables) [13](#)
  - DSQTSOBJ (SAVE DATA results) [13](#)
- databases
  - DSQDBCTL (QMF control tables) [13](#)
  - DSQDBDEF (SAVE DATA results) [13](#)
- DSQUOPTS routine [132](#)
- execs
  - DSQ1DEFS (installation parameter defaults) [40](#)
  - DSQSCMDn (stored procedure interface parameters) [107](#)
- exit routines
  - DSQUOPTS (initializes global variables) [131](#)
- global variables
  - DSQAO\_OTC\_LICENSE [77](#)
  - DSQAO\_OTC\_LICENSE (QMF VUE licenses) [132](#)
  - DSQAO\_QMF\_VER\_RLS (current QMF release) [77](#)
  - DSQAO\_STO\_PROC\_INT (stored procedure interface) [107](#)
  - DSQEC\_CC (carriage control suppression) [119](#)
  - DSQEC\_DISABLEADM (QMF administrator authority checking) [132](#)
  - DSQEC\_ISOLATION (transaction isolation level) [189](#)
  - DSQEC\_LAST\_RUN (last used date in object lists) [132](#)
  - DSQEC\_SHARE (SHARE parameter default) [132](#)
- installation jobs
  - DSQ1BFRM (CICS charts and trace) [58](#)
  - DSQ1BINR (binds QMF application plan) [45](#), [47](#)
  - DSQ1BLNI (creates V12.1 control tables) [45](#), [62](#)
  - DSQ1BPKG (binds QMF packages) [30](#), [45](#), [47](#), [62](#), [64](#), [66](#), [68](#)
  - DSQ1BSP (stored procedure interface) [107](#)
  - DSQ1BSQL (binds installation programs) [30](#), [45](#), [47](#), [62](#), [64](#), [66](#), [68](#)
  - DSQ1BUDF (installs enhanced LIST command) [45](#), [62](#), [109](#)
  - DSQ1BUDV (enhanced LIST command views) [30](#), [45](#), [62](#), [109](#)
  - DSQ1BVW (installs default LIST command views) [30](#), [45](#), [62](#), [66](#), [109](#)
  - DSQ1CFRM (CICS charts and trace) [58](#)
  - DSQ1EADM (GDDM map groups) [58](#)
  - DSQ1EAS4 (creates sample tables on iSeries) [64](#)
  - DSQ1ECSO (CICS system definition) [56](#)
  - DSQ1EDJ4 (creates sample tables on LUW) [62](#), [64](#)
  - DSQ1EDSJ (deletes sample tables on z/OS) [47](#), [64](#)
  - DSQ1EDX2 (deletes sample tables on iSeries, LUW) [64](#)
  - DSQ1EGLK (CICS governor preparation) [56](#)
  - DSQ1EIVQ (creates QMF Analytics for TSO sample tables) [62](#)
  - DSQ1EIVS (creates sample tables on z/OS) [45](#), [47](#), [62](#), [64](#)
  - DSQ1EPNL (populates panel library) [37](#), [75](#)
  - DSQ1ESQD (deletes sample queries/procedures) [77](#)
  - DSQ1ESQI (installs sample queries/procedures) [77](#)
  - DSQ1nCCS (creates NLF command synonyms) [101](#)
  - DSQ1nRTS (creates new command synonyms) [106](#)
  - DSQ1STGJ (creates storage spaces) [45](#), [62](#)
  - DSQ1TBAJ (creates VCAT name) [45](#), [62](#)



DSQ, names that begin with (*continued*)

- installation jobs (*continued*)
  - DSQ1UOPT (assembles, link-edits DSQUOPTS exit routine) [132](#)
- load modules
  - listings [38](#)
- package names
  - release identifiers [91](#)
- procedures
  - DSQ0BINI (default system initialization procedure) [137](#)
  - DSQ1EBAT (TSO installation verification, batch) [81](#)
  - DSQ1EIVC (CICS installation verification) [82](#)
  - DSQ1EIVP (TSO installation verification) [80](#)
  - TSO logon [50](#)
- program parameters
  - DSQSBSTG [151](#)
  - DSQSBSTG (fixed virtual storage) [151](#)
  - DSQSDBCS (printing DBCS data) [169](#)
  - DSQSDBLG (trace log destination, stored procedure interface) [119](#)
  - DSQSDBNM (initial database location) [6](#), [149](#)
  - DSQSDBQN (trace queue name in CICS) [168](#)
  - DSQSDBQT (trace queue type in CICS) [168](#)
  - DSQSDEBUG (trace level) [167](#)
  - DSQSIROW (rows fetched before report display) [159](#)
  - DSQSMODE (batch vs. interactive operation) [162](#)
  - DSQSMRFI (multirow fetch/insert) [160](#)
  - DSQSPILL (spill storage for report operations) [152](#)
  - DSQSPLAN (QMF application plan ID) [149](#)
  - DSQSPRID (QMF profile key) [150](#)
  - DSQSPTYP (extended storage for spill data in TSO) [152](#)
  - DSQSRSTG (dynamic virtual storage) [151](#)
  - DSQSRUN (initial procedure name) [163](#)
  - DSQSSPQN (CICS queue name for spill data) [157](#)
  - DSQSSUBS (database subsystem name) [6](#)
  - DSQSSUBS (Db2 subsystem ID) [148](#)
- programs
  - DSQQMFE [72](#), [74](#), [117](#), [118](#)
- queues
  - DSQD [75](#)
- starting QMF
  - initializing global variables in DSQUOPTS [132](#)
- stored procedures
  - DSQABA1E (enhanced LIST command) [109](#), [402](#)
  - DSQQMFSP (QMF stored procedure interface) [107](#)
- table spaces
  - DSQTSCT1 (Q.OBJECT\_DIRECTORY control table) [40](#)
  - DSQTSCT2 (Q.OBJECT\_REMARKS control table) [40](#)
  - DSQTSCT3 (Q.OBJECT\_DATA control table) [40](#)
  - DSQTSDEF (SAVE DATA results) [13](#), [40](#)
  - DSQTSGOV (Q.RESOURCE\_TABLE control table) [40](#)
  - DSQTSLOG (Q.ERROR\_LOG control table) [40](#)
  - DSQTSPRO (Q.PROFILES control table) [40](#)
  - DSQTSRDO (Q.DSQ\_RESERVED control table) [40](#)
  - DSQTSSTYN (Q.COMMAND\_SYNONYMS control table) [40](#)
- uninstall jobs
  - DSQ1DEL1 (frees application plan on z/OS) [89](#), [92](#)
  - DSQ1DEL2 (drops database constructs on z/OS) [92](#)

DSQ, names that begin with (*continued*)

- uninstall jobs (*continued*)
  - DSQ1DEL3 (deletes user-managed data sets on z/OS) [92](#)
  - DSQ1DELA (deletes prior QMF release, z/OS) [92](#)
  - DSQ1EDX1 (drops control tables/packages on iSeries, LUW) [94](#)
  - DSQ1EDX2 (deletes sample tables on iSeries, LUW) [95](#)
  - DSQ1JFPL (frees application plan on z/OS) [89](#)
- DSQ10297 [340](#)
- DSQ10493 [340](#)
- DSQ1DEFS exec
  - SECAUTH parameter [13](#)
- DSQ1EGLK [338](#)
- DSQAO\_STO\_PROC\_INT global variable [107](#)
- DSQDBCTL database [13](#)
- DSQDBDEF database [13](#)
- DSQDC\_POS\_SQLCODE variable [354](#)
- DSQDEBUG data set
  - stored procedure interface [119](#)
  - stored procedure interface, allocation [107](#)
  - support for positive SQL codes [354](#)
  - under CICS [356](#)
- DSQEC\_BUFFER\_SIZE global variable [351](#)
- DSQEC\_CC global variable
  - stored procedure interface [119](#)
- DSQEC\_CON\_ACC\_RES global variable [351](#)
- DSQEC\_CON\_CSWL global variable [352](#)
- DSQEC\_DISABLEADM global variable [131](#)
- DSQEC\_EXTND\_STG global variable [152](#)
- DSQI0026 [340](#)
- DSQPNLE
  - CSD definition [338](#)
- DSQQMFE [338](#)
- DSQQMFSP procedure, *See* stored procedure, starting QMF for TSO as
- DSQSCMDn exec
  - stored procedure interface [107](#)
- DSQTSCTL database partition group [13](#)
- DSQTSDEF table space [13](#)
- DSQTSOBJ database partition group [13](#)
- DSQUCFRM ddname [225](#)
- DSQUECIC edit program [265](#), [270](#), [277](#)
- DSQUEGV1 module, governor exit [294](#)
- DSQUEGV3 module, governor exit [294](#)
- DSQUOPTS user exit
  - description [131](#)
- dumps for diagnosis [364](#)
- DXEGOVA control block [299](#)
- DXEXCBA control block [303](#)
- dynamic statement cache search
  - improving reuse [352](#)

## E

- E trace code [353](#)
- edit
  - codes
    - CASE field of profile [258](#)
    - numeric data processing [258](#)
  - exit interface
    - control block fields [254](#)
    - DBCS support [259](#)

- edit (*continued*)
  - exit interface (*continued*)
    - input area [253, 254](#)
    - output area [253, 254](#)
    - termination calls [253](#)
  - routine [253](#)
- EDIT TABLE command
  - SQL privileges required [185](#)
- EDSA limit size (CICS) for large QMF queries [60](#)
- enhancements
  - to prior releases [367](#)
- ENQ command
  - print queues [221](#)
- ENTRY\_TYPE column (function key table) [247](#)
- environment
  - customizing [172](#)
  - default setup [405](#)
- ENVIRONMENT column (Q.PROFILES)
  - role in profile initialization [181](#)
- ERASE command
  - QMF administrator privileges [13](#)
- error
  - initialization [340](#)
- errors, *See* troubleshooting
- EXECUTE privilege
  - access, QMF application plan and packages [171](#)
- EXPLAIN
  - enabling in QMF [347](#)
- EXPORT command
  - date last used, setting behavior [201](#)
  - QMF administrator privileges [13](#)
- EXPORT TABLE command
  - governing [315](#)
  - SQL privileges [185](#)
- extended floating point, edit routine [258](#)
- extended storage for report operations (TSO) [152, 349](#)
- extended-format decimal floating point, *See* DECFLOAT data type

## F

- F trace code [353](#)
- fallback to prior QMF release
  - backward compatibility considerations [85](#)
  - procedure for fallback to CM release
    - servers [67](#)
- Family 1 printer [214, 217](#)
- Family 2 printer [214, 217](#)
- Family 3 printer [214, 217](#)
- Family 4 printer [214](#)
- feedback, sending to IBM [xiii](#)
- fetch program parameters
  - multirow fetch [160](#)
  - number of report rows retrieved [159](#)
- FLOAT data type
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- FLOAT(DFP) compiler option, user edit routines [268, 270](#)
- floating point data, edit routine [258](#)
- floating-point numbers
  - importing single-precision [383](#)
- folders
  - internal stored format [204](#)
  - listing [207](#)

- forms
  - creating new edit codes [258](#)
  - displaying [208](#)
  - internal stored format [204](#)
  - listing [207](#)
  - print preview [228](#)
  - printing [223](#)
  - window IDs [240](#)
- FORWARD command
  - governing [315](#)
- forward compatibility between QMF releases [85](#)
- FSFRCE services, printing [214](#)
- full-screen panels
  - customized function key examples [249](#)
  - panel IDs [239](#)
  - tracing [353](#)
- function calls
  - branch addresses [297](#)
  - types [291](#)
- function keys
  - customizing
    - activating new definitions [248](#)
    - problems activating [246](#)
    - updating function key table [246](#)
  - index on table [245](#)
  - table
    - creating [245](#)
    - entering definitions [246](#)
- function-level trace, configuring [353](#)

## G

- G trace code [353](#)
- G050 abend [338](#)
- GDDM (Graphical Data Display Manager)
  - add maps to ADMF data set [101](#)
  - ADMADFT defaults module [218](#)
  - APPCPG parameter [49, 57](#)
  - common errors on QMF startup [338](#)
  - display devices supported [33](#)
  - error messages, printing [345](#)
  - printer nicknames
    - ADMMNICK specification [217](#)
  - printing [214](#)
- GDDM-PGF [406](#)
- global variable table
  - structure [135](#)
- global variables
  - DSQAO\_STO\_PROC\_INT [107](#)
  - DSQDC\_POS\_SQLCODE [354](#)
  - DSQEC\_BUFFER\_SIZE [351](#)
  - DSQEC\_CC, stored procedure interface [119](#)
  - DSQEC\_CON\_ACC\_RES [351](#)
  - DSQEC\_CON\_CSWL [352](#)
  - DSQEC\_DISABLEADM [131](#)
  - DSQEC\_EXTND\_STG [152](#)
  - DSQEC\_LAST\_RUN [132](#)
  - initializing when QMF starts [131](#)
  - printing [221](#)
  - Q.GLOBAL\_VARS table
    - setting values [134](#)
  - setting in Q.GLOBAL\_VARS table [134](#)
  - starting QMF
    - initializing global variables in Q.GLOBAL\_VARS [134](#)

- global variables (*continued*)
  - window IDs [240](#)
- governor exit routine
  - cancellation service [313](#)
  - CICS control block interface [298](#)
  - command processing [294](#)
  - comparison with Db2 resource limit facility, *See* resource limit facility, Db2
  - control blocks
    - DXEGOVA [299](#)
    - DXEXCBA [303](#)
  - control information, storing [310](#)
  - description [281](#)
  - entry point [290](#)
  - exit routine information [303](#)
  - flow of control [298](#)
  - function calls [297](#)
  - performance [297](#)
  - program structure [298](#)
  - resource control table [281](#)
  - scratchpad area [310](#)
  - specifying for resource groups [286](#)
  - types of function calls [291](#)
- governor exit routine QMF
  - passing resource control information [298](#)
- governor, Db2, *See* resource limit facility, Db2
- granting to PUBLIC AT ALL LOCATIONS [187](#)
- GRAPHIC data type
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- graphics printers, defining nicknames [214](#)

## H

- HANDLE CONDITION
  - CICS [294](#)
- hardware prerequisites [33](#)
- help panels, customizing keys [240](#), [249](#)
- hex edit codes [347](#)
- HEX function [347](#)
- High Performance Option/Manager [281](#)
- highlighting conventions [xii](#)
- HPO/Manager [281](#)

## I

- I trace code [353](#)
- implicitly created table spaces [198](#)
- IMPORT command
  - date last used, setting behavior [201](#)
  - QMF administrator privileges [13](#)
  - single-precision floating point support [383](#)
  - storage, *See* allocation of storage
- IMPORT TABLE command
  - creating tables [197](#)
  - governing [315](#)
  - SQL privileges required [185](#)
- indexes
  - Q.PROFILES table [172](#)
  - recreating [205](#)
- initial procedure
  - receiving variable values [165](#)
- initialization

- initialization (*continued*)
  - authentication process [107](#), [150](#), [181](#)
  - global variable values (DSQUOPTS routine) [131](#)
  - message numbers [353](#)
  - performance [405](#)
  - QMF administrator authority verification
    - disabling [131](#)
    - process [13](#)
  - storage requirements [37](#)
  - troubleshooting [340](#)
- input area
  - control for formatting [253](#)
  - control for termination [253](#)
- installation
  - authority required [13](#)
  - cleanup [89](#)
  - coexistence of QMF releases [4](#)
  - database releases supported [3](#)
  - determining current installation mode [14](#)
  - information roadmaps, *See* installation roadmaps
  - migrating objects and programs [85](#)
  - modes in prior releases [4](#)
  - National Language Features (NLFs) [101](#)
  - objects required for remote unit of work [7](#)
  - overview [11](#)
  - planning, *See* planning for installation
  - prerequisites [33](#)
  - resources created [13](#)
  - restrictions [5](#)
  - server database installs [61](#)
  - stored procedure interface (TSO) [107](#)
  - testing and verification [71](#)
- installation packages
  - required locations [7](#)
- installation plan
  - required locations [7](#)
- installation roadmaps
  - Db2 for z/OS requesters
    - new QMF installations [16](#)
    - overview [14](#)
  - National Language Features [29](#)
  - overview [11](#)
  - three-part name access [27](#)
- installation verification procedure
  - requester databases [71](#)
  - server databases [83](#)
  - stored procedure interface [107](#)
- INTEGER data type
  - edit routines [254](#), [258](#)
  - width [153](#)
- intercepts for DCB abends, spill file considerations [349](#)
- interface control block
  - DXEGOVA [298](#)
  - DXEXCBA [298](#)
- interrelease compatibility [85](#)
- interrupt facility
  - using [354](#)
- invalid subsystem ID [340](#)
- ISC (intersystem communication) [406](#)
- iSeries platform
  - authorities required [13](#)
  - database prerequisites [3](#)
  - distributed unit of work configuration [8](#)
  - remote unit of work configuration [5](#)

isolation levels  
  cursor stability [189](#)  
  uncommitted read [189](#)  
ISPF (Interactive System Productivity Facility)  
  common error [340](#)  
  LIBDEF allocations for QMF libraries [126](#)

## J

JCL  
  stored procedure interface, QMF for TSO [107](#)  
job control language, *See* JCL  
jobs that install QMF  
  requester databases  
    new V12.1 installations [45](#)  
  server databases  
    new Version 12.1 installations [62](#)  
    three-part name access [68](#)

## K

keywords, reporting problems [365](#)

## L

L2 trace option  
  stored procedure interface [119](#)  
language features of QMF, *See* National Language Feature (NLF)  
language parameter, stored procedure interface [119](#)  
last used date on objects [132](#), [201](#)  
LAYOUT command  
  date last used, setting behavior [201](#)  
LEFT command  
  governing [315](#)  
legal  
  notices [407](#)  
  trademarks [408](#)  
LENGTH column (Q.PROFILES) [172](#)  
LIBDEF statements for QMF library allocations [126](#)  
license agreement for QMF VUE [16](#), [18](#)  
LIKE keyword  
  how support varies by database [383](#)  
linear procedures in command synonyms [231](#)  
link-edit statements  
  governor exit routine [311](#), [312](#)  
Linux, *See* Linux, UNIX, and Windows platform  
Linux, UNIX, and Windows platform  
  authorities required [13](#)  
  database prerequisites [3](#)  
  distributed unit of work configuration [8](#)  
  packages available for governing [315](#)  
  remote unit of work configuration [5](#)  
LIST command  
  ALL keyword [208](#)  
  listing tables authorized to secondary IDs  
    after upgrading Db2 [30](#)  
    installing feature that allows [109](#)  
list views  
  creating [190](#)  
  secondary authorization IDs [109](#)  
literals in command synonyms [233](#)  
load modules

load modules (*continued*)  
  moving to enhance performance [38](#)  
LOB data  
  storage objects [390](#)  
location window IDs [240](#)  
locks on data  
  uncommitted read vs. cursor stability [189](#)  
logon to QMF  
  enabling [172](#)  
  restricting [180](#)  
LONG VARCHAR data type  
  SQLTYPE, user edit routines [254](#)  
  width [153](#)  
LONG VARGRAPHIC data type  
  SQLTYPE, user edit routines [254](#)  
  width [153](#)  
long-format decimal floating point, *See* DECFLOAT data type  
LUW, *See* Linux, UNIX, and Windows platform

## M

machine requirements for QMF [33](#)  
maintenance  
  command synonym table [237](#)  
  displaying objects [208](#)  
  enlarging table space for objects [205](#)  
  listing objects [207](#)  
  QMF administrator authority [13](#)  
messages  
  canceling user activity, governor [310](#)  
  diagnosing problems [353](#)  
  number ranges and what they mean [353](#)  
  row limit exceeded [284](#)  
  status, stored procedure interface [119](#)  
  system error messages [354](#)  
  tracing  
    stored procedure interface [119](#)  
    support for positive SQL codes [354](#)  
migration  
  National Language Features (NLFs) [106](#)  
migration of QMF  
  coexistence of QMF releases [4](#)  
  database releases supported [3](#)  
  database-only upgrade tasks [30](#)  
MODEL column [201](#)  
MODEL column (Q.PROFILES) [172](#)  
modes of installation in prior QMF releases [4](#)  
modules for QMF program, *See* load modules  
MRO (multiregion operation) [406](#)  
multiple database threads [161](#)  
multirow fetch  
  program parameter [160](#)  
  three-part name restrictions [8](#)  
multistatement queries  
  restrictions [171](#)  
  units of work [5](#)

## N

names  
  ADMMNICK specification [217](#)  
  objects  
    supported lengths [4](#)

- names (*continued*)
  - three-part, multirow fetch [160](#)
- National Language Feature (NLF)
  - configuring stored procedure interface [107](#)
  - DBCS device requirements [33](#)
  - installation
    - procedure [101](#)
    - roadmap [29](#)
  - migration
    - procedure [106](#)
  - specifying in stored procedure interface [119](#)
- networks and QMF configuration [5](#)
- New Function Mode
  - checking for [14](#), [19](#)
  - Db2 for z/OS
    - QMF installation requirements [3](#)
  - QMF
    - coexistence with Compatibility Mode [4](#)
    - prior releases [4](#)
- nickname
  - defined [214](#)
  - defining multiple printers [217](#)
  - errors during printing [345](#)
- NLF (National Language Feature)
  - command synonyms [231](#)
  - release numbers [365](#)
  - See also* National Language Feature (NLF)
- notices
  - legal [407](#)
- NUMBER column (function key table) [247](#)
- numeric
  - data
    - conversion, edit routine [258](#)
    - importing single-precision floating point [383](#)
- NUMTCB parameter, stored procedure interface [107](#)

## O

- object
  - control tables [201](#)
  - date last used [132](#), [201](#)
  - deleting [209](#)
  - displaying [208](#)
  - internal representation [201](#)
  - list
    - customizing [190](#)
    - window IDs [240](#)
  - listing [207](#)
  - maintenance [201](#)
  - name, command synonym [229](#)
  - names
    - supported lengths [4](#)
  - sharing [208](#)
- object catalog
  - coexistence of QMF releases [4](#)
  - compatibility between QMF releases [4](#)
  - DSQDBCTL database [13](#)
- OBJECT column (synonyms table) [230](#)
- object lists
  - customizing with global variables [190](#)
- OBJECTLEVEL column, QMF control tables [201](#)
- online help
  - message support
    - setting handling of positive SQL codes [354](#)

- OTC\_LICENSE variable (QMF VUE license agreement) [132](#)
- output area
  - control for formatting [253](#)
  - control for termination [253](#)
- OWNER column, QMF control tables [201](#)
- ownership
  - how QMF tracks [201](#)
  - transferring [208](#)

## P

- P trace code [353](#)
- packages, QMF
  - governing with Db2 resource limit facility [315](#)
- PANEL column (function key table) [246](#)
- panels
  - class ID [246](#)
  - governor prompt [284](#)
  - IDs [239](#), [240](#)
  - print and display support [406](#)
- parameters
  - passed to edit routine [253](#)
  - passing [141](#)
- partition groups, database, *See* databases, partition groups
- PC printers [217](#)
- performance
  - DSQSIROW, large values [159](#)
  - DSQSIROW, small values [159](#)
  - managing [347](#)
  - resident programs [405](#)
  - table indexes [197](#)
  - troubleshooting [349](#), [351](#), [352](#)
  - using spill file [157](#)
- performance, improving with DSQEC\_BUFFER\_SIZE [351](#)
- PF\_SETTING column (function key table) [247](#)
- PFKEYS column (Q.PROFILES) [172](#)
- planning for installation
  - coexistence of QMF releases [4](#)
  - distributed data configurations
    - distributed unit of work [8](#)
    - remote unit of work [5](#)
  - planning for installation
    - installation modes [4](#)
  - prerequisites [33](#)
  - roadmaps for installation [11](#)
- post-installation cleanup
  - base QMF (English) [89](#)
  - NLF [101](#)
- prerequisites for installation
  - authority [12](#)
  - concepts [12](#)
  - database releases [3](#)
  - hardware [33](#)
  - software [33](#)
- previewing printed reports [228](#)
- previous QMF releases
  - deleting [89](#)
  - Version 12.1 compatibility [85](#)
- PRINT command
  - date last used, setting behavior [201](#)
  - routing to named destinations [213](#), [221](#)
- print preview function [228](#)
- PRINT TABLE command
  - governing [315](#)

PRINT TABLE command (*continued*)  
 SQL privileges required [185](#)

printer  
 ANSI support  
 graphic device [213](#)  
 control keywords (PRINTCTL) [217](#)  
 multiple addresses [217](#)  
 nicknames (GDDM) [340](#)

PRINTER column (Q.PROFILES) [119](#), [172](#)

printing  
 enabling users [213](#)  
 errors [345](#)  
 QMF vs. GEM [213](#)  
 using GDDM services [214](#)

prior QMF releases  
 deleting [89](#)  
 Version 12.1 compatibility [85](#)

privileges  
 database objects [185](#)  
 QMF administrator authority [13](#)  
 required for QMF tasks [185](#)  
 revoking grants of others [188](#)  
 STATS and REORG [209](#)  
 stored procedure interface [107](#)  
 See also table privilege

problem reporting [365](#)

problems, See troubleshooting

procedures  
 displaying [208](#)  
 internal stored format [204](#)  
 listing [207](#)  
 maintaining objects [204](#)  
 printing [223](#)  
 running through stored procedure interface [107](#)  
 using in command synonyms [231](#)

processor time  
 controlling use [285](#)  
 setting limits [281](#)

processors supported for QMF [33](#)

PROCOPT parameter, printing [217](#)

profile  
 authentication during logon [107](#), [181](#)  
 CASE setting, customized function keys [247](#)  
 command synonyms [236](#)  
 creating [172](#), [180](#)  
 default settings [180](#)  
 default values [172](#)  
 deleting [180](#), [184](#)  
 initialization search order [181](#)  
 maintenance [201](#)  
 printing [223](#)  
 SET PROFILE command [183](#)  
 trace values  
 stored procedure interface [119](#)  
 updating [183](#), [184](#)

PROFILE PREFIX statement [320](#)

PROG 759 [340](#)

program parameters  
 DSQSBSTG [151](#)  
 DSQSDBCS [169](#)  
 DSQSDBLG [119](#)  
 DSQSDBNM [6](#), [149](#)  
 DSQSIROW [159](#)  
 DSQSMODE [162](#)

program parameters (*continued*)  
 DSQSMRFI [160](#)  
 DSQSPILL [141](#), [152](#)  
 DSQSPLAN [149](#)  
 DSQSPRID [150](#)  
 DSQSPTYP [141](#)  
 DSQSRSTG [151](#)  
 DSQSRUN [163](#)  
 DSQSSPQN [157](#)  
 DSQSSUBS [6](#), [148](#)  
 stored procedure interface (TSO) [107](#)  
 summary [141](#)  
 trace  
 DSQSDBQN [168](#)  
 DSQSDBQT [168](#)  
 DSQSDBUG [167](#)

programming interface information [408](#)

prompt panels  
 customized function key example [249](#)  
 panel IDs [240](#)

prompted query  
 printing [214](#), [223](#)  
 SQL privileges [185](#)  
 window IDs [240](#)

PSP subset names for QMF [335](#)

PTF trace option (stored procedure interface) [119](#)

PTFs for QMF, applying [335](#)

PUBLIC, granting to, See granting to PUBLIC AT ALL LOCATIONS

## Q

Q authorization ID [13](#)

Q.COMMAND\_SYNONYMS\_n table  
 job to create [101](#)

Q.ERROR\_LOG control table [362](#)

Q.GLOBAL\_VARS table  
 structure [135](#)

Q.OBJECT\_DIRECTORY  
 LAST\_USED column [132](#), [201](#)

Q.PROFILES control table  
 adding user profiles [181](#)  
 administrator authority checking [13](#)  
 deleting user profile [180](#)  
 table structure [172](#)  
 update for NLF [101](#)  
 updating [183](#)  
 updating RESOURCE\_GROUP field [285](#)  
 user authentication [181](#)  
 user modifications [183](#)

Q.RESOURCE\_VIEW, governor [284](#)

QBE queries  
 governing [315](#)  
 printing [223](#)  
 SQL privileges [185](#)

QMF administrator authority  
 disabling [131](#)  
 verification process and privileges [13](#)

QMF Analytics for TSO  
 sample tables [99](#)

query  
 created during installation [13](#)  
 deleting [209](#)  
 displaying [208](#)



query (*continued*)  
  internal stored format [204](#)  
  listing [207](#)  
  multistatement, *See* multistatement queries  
  running through stored procedure interface [107](#)  
  statement length [383](#)  
query acceleration  
  enabling [348](#)  
query accelerator [348](#)  
QUEUENAME, QUEUETYPE keywords [221](#)

## R

RACF  
  batch security [319](#)  
  group names, *See* secondary authorization IDs  
  installation considerations [13](#)  
RCT (Resource Control Table) [340](#)  
release numbers for NLFs [365](#)  
REMARKS column [205](#)  
remote data access  
  overview [5](#)  
remote Db2 client  
  QMF batch jobs [332](#)  
remote unit of work  
  creating command synonym tables [117](#)  
  customizing a remote database connection [117](#)  
  DB2 for VSE and VM default views [192](#)  
  DRDA requirements [5](#)  
  installation roadmaps  
    non-z/OS servers [19](#)  
    z/OS-only servers [14](#)  
  multirow fetch with [160](#)  
  required configuration [5](#)  
  restrictions [6](#)  
  server installations [62](#)  
removing Db2 privileges  
  incomplete revocations [188](#)  
  the cascade effect [188](#)  
report formatting services, tracing [353](#)  
reports  
  binary data [347](#)  
  data formats [253](#)  
  print preview [228](#)  
  printing [223](#)  
  Q.ERROR\_LOG table [362](#)  
  receiving in result sets (stored procedure interface) [119](#)  
  slow performance [349](#)  
  storage  
    dynamic allocation [151](#)  
    fixed allocation [151](#)  
    percentage allocation [151](#)  
    spill data [152](#)  
    troubleshooting [349](#)  
    variable allocation [151](#)  
requester databases  
  configuration  
    overview [5](#)  
  installations  
    install job parameters [40](#)  
    jobs that install a new release [45](#)  
    jobs that migrate a prior release [47](#)  
    loading QMF Version 12.1 libraries [34](#)  
    moving modules [38](#)

requester databases (*continued*)  
  installations (*continued*)  
    prerequisites [33](#)  
    roadmaps [14](#)  
    storage requirements [37](#)  
    TSO and CICS preferences [49](#)  
requirements for installation  
  authority [12](#)  
  concepts [12](#)  
  database releases [3](#)  
  hardware [33](#)  
  software [33](#)  
resident QMF programs [405](#)  
resource  
  group [180](#)  
  profile management [180](#)  
resource contention  
  concurrent access resolution options [351](#)  
  *See also* contention for resources  
resource limit facility, Db2  
  comparison with QMF governor [315](#)  
  QMF commands that can be governed [315](#)  
resource limit specification table (Db2) [317](#)  
RESTRICTED column  
  changing value to NO [208](#)  
  defined [201](#)  
result sets, receiving reports as [119](#)  
REVOKE statements  
  example [188](#)  
REXX support  
  QMF for TSO stored procedure interface [107](#)  
RIGHT command  
  governing [315](#)  
RLSTs [317](#)  
roadmaps for installation  
  CONNECT command and three-part name access [14, 19](#)  
  distributed unit of work configuration [9](#)  
  general installation steps [11](#)  
  multicultural support [29](#)  
  remote unit of work configuration [7](#)  
  requester databases (Db2 for z/OS)  
    new QMF installations [16](#)  
    overview [14](#)  
  server databases  
    overview [19](#)  
    three-part name access [27](#)  
  stand-alone configuration [14](#)  
  three-part name access only [27](#)  
rows, database  
  controlling number retrieved [284](#)  
rules  
  command synonyms [229](#)  
  customizing function keys [246](#)  
RUN command  
  command synonym [231](#)  
  date last used, setting behavior [201](#)  
  SQL privileges required [185](#)  
  storage, *See* allocation of storage  
RUN QUERY command  
  governing [315](#)

## S

sample tables

- sample tables (*continued*)
  - creating
    - NLF [101](#)
    - NLF (QMF Analytics for TSO) [106](#)
  - deleting
    - NLF [101](#)
  - installing NLF [101](#)
  - QMF Analytics for TSO [99](#)
- SAVE command
  - DATA keyword [185](#)
  - date last used, setting behavior [201](#)
  - enhancement [198](#)
  - QMF administrator privileges [13](#)
  - SQL privileges required [185](#)
  - TABLE keyword [185](#)
- SAVE DATA command
  - governing [315](#)
  - LOB storage [390](#)
  - receiving table space [13](#)
  - storage, *See* allocation of storage
- SAVE option
  - EDIT TABLE command
    - unsupported situations [383](#)
- SBCS languages, CCSIDs [49](#), [57](#)
- SCOPE resource option [282](#)
- scratchpad area
  - governor exit routine [310](#)
- SECAUTH installation parameter [13](#)
- secondary authorization IDs
  - installation with [13](#)
  - LIST command support [109](#)
- security
  - administrator authority verification, *See* QMF administrator authority
  - RACF, *See* RACF
- SELECT statement
  - restrictions for multistatement queries [171](#)
- SEQ column [204](#)
- server databases
  - configuration
    - overview [5](#)
  - connecting during initialization [6](#)
  - installations
    - CONNECT command access [19](#)
    - install job parameters [40](#)
    - minimum database versions [3](#)
    - three-part name access [27](#)
- service
  - PSP subset names for QMF [335](#)
- service information [xii](#)
- session [172](#)
- SET PROFILE command [183](#)
- share locks on data, *See* locks on data
- SHARE parameter, SAVE command [13](#)
- shift characters [259](#)
- single-precision floating point numbers
  - support [383](#)
- slow performance [349](#)
- small integer data, edit routine [258](#)
- SMALLINT data type
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- software prerequisites [33](#)
- Software Support [365](#)
- SPACE column (Q.PROFILES) [172](#)
- special registers for CCSID values [49](#), [57](#)
- spill data
  - extended storage (TSO only) [152](#)
  - file
    - CICS [157](#)
    - TSO [153](#)
- spill file
  - DCB abend intercept considerations [349](#)
  - estimating size [153](#)
  - performance problems [157](#)
  - sample calculations [153](#)
- SQL
  - HEX function [347](#)
  - ID
    - command synonym table [238](#)
    - how QMF stores [204](#)
  - privileges
    - for prompted, QBE queries [185](#)
    - for QMF commands [185](#)
  - return codes [354](#)
  - statement
    - INSERT (new user profile) [181](#)
    - UPDATE [183](#)
- SQL codes
  - 551 [131](#), [340](#)
  - 805 [340](#)
  - message support [354](#)
  - setting handling of positive [354](#)
- SQL queries
  - larger than 32 KB
    - governing [289](#)
- SQL statements
  - governing through QMF with Db2 resource limit facility [315](#)
- SQLTYPEs of data in user edit routines [254](#)
- stand-alone database configuration
  - installation roadmaps [14](#)
- starting QMF
  - common errors [338](#)
  - initializing global variables [131](#)
  - QMF administrator authority overrides [131](#)
  - QMF profile initialization [181](#)
  - stored procedure interface [107](#), [119](#)
- statement, query
  - lengths supported [383](#)
- storage
  - data from edit routine [253](#)
  - moving modules to enhance [38](#)
  - report operations
    - program parameters [150](#)
    - spill data [152](#), [349](#)
  - requirements
    - initialization [37](#)
    - region size [107](#)
    - report operations [37](#)
  - table space, increasing size [205](#)
  - troubleshooting [349](#)
- storage-related issues
  - resolving [347](#)
- stored procedure
  - starting QMF for TSO as
    - authentication [107](#)
    - installing interface [107](#)



- stored procedure (*continued*)
  - starting QMF for TSO as (*continued*)
    - restrictions [8](#)
  - starting TSO as
    - restrictions [6](#)
- subsystem ID
  - specifying, remote unit of work [6](#)
- SUBTYPE column, QMF control tables [201](#)
- summary of changes [367](#)
- support information [xii](#)
- SUSPEND keyword [221](#)
- SYNONYM\_DEFINITION column [231](#)
- SYNONYMS column (Q.PROFILES) [172](#)
- synonyms for QMF commands
  - creating synonyms table [234](#)
  - initialization messages [340](#), [344](#)
  - object name [229](#)
  - problems activating [229](#)
  - quotation marks [233](#)
  - synonym definition [231](#)
  - syntax [233](#)
  - table maintenance [237](#)
  - using variables [232](#)
  - verb [229](#)
- syntax diagrams
  - CALL statement [107](#)
- syntax diagrams, how to read [xii](#)
- SYSADM authority
  - installation prerequisites [13](#)
  - REVOKE statements [188](#)
  - revoking access to the application plan [171](#)
- SYSIBM.SYSTEM\_EBCDIC\_CCSID session variable [49](#), [57](#)
- system initialization procedure, default [137](#)
- SYSTEM profile
  - changing default values [184](#)
  - deleting [180](#)
- SYSTEM row of QMF profile
  - authentication using [107](#)
- systems interface, tracing [353](#)
- SYSTSPRT data set
  - stored procedure interface, allocation [107](#)

## T

- Table Editor
  - SQL privileges required [185](#)
- table privilege
  - overview [187](#)
- table space
  - assigning [198](#)
  - creating tables [197](#)
  - default space, SAVE DATA command [13](#)
  - deleting [184](#)
  - enlarging [205](#)
  - explicit/implicit option
    - CREATE TABLE statement [199](#)
    - SAVE and IMPORT commands [198](#)
  - explicitly created table spaces [198](#)
  - implicitly created table spaces [198](#)
  - specifying in user profile [172](#)
- tables
  - controlling access [185](#)
  - created during installation [13](#)
  - function keys [239](#)

- tables (*continued*)
  - indexes [197](#)
  - printing [223](#)
  - resource control, governor exit [282](#)
  - SAVE DATA configuration [13](#)
- temporary storage queue
  - printing using QMF services [221](#)
- termination calls, edit routine [253](#)
- termination messages [364](#)
- terms of license agreement, QMF VUE [16](#), [18](#)
- testing installation
  - requester databases [71](#)
  - server databases [83](#)
  - stored procedure interface [107](#)
- three-part names
  - database support [383](#)
  - governing remote access [315](#)
  - installation path
    - jobs [28](#)
    - roadmap [27](#)
    - T3PARTNM parameter [61](#)
  - installation paths
    - jobs [68](#)
  - installation roadmaps [9](#)
  - objects accessed [9](#)
  - required configuration [8](#), [9](#)
  - restrictions
    - multirow fetch/insert [160](#)
    - VM/VSE platforms [8](#)
- TIME data type
  - edit routines that format [261](#)
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- timeout, QMF transaction
  - CICS region size [349](#)
  - defining message display [364](#)
- TIMESTAMP data type
  - edit routines that format [261](#)
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- TIMESTAMP WITH TIME ZONE data type
  - edit routines that format [261](#)
  - SQLTYPE, user edit routines [254](#)
  - width [153](#)
- TOFAM keyword, ADMMNICK specification [217](#)
- toggle switch, governor exit [282](#)
- TOP command
  - governing [315](#)
- TRACE column (Q.PROFILES) [172](#)
- tracing problems
  - allocation of output
    - CICS [58](#), [75](#)
    - TSO [50](#)
  - batch mode procedures [326](#), [331](#), [334](#)
  - CICS
    - allocation of output [141](#)
    - diagnosis aids [364](#)
  - command synonyms [234](#)
  - creating interrupts [354](#)
  - data
    - allocating data set [356](#)
    - viewing [360](#)
  - DSQDEBUG data set [50](#), [58](#), [107](#)
  - error log reports [362](#)

- tracing problems (*continued*)
    - facility
      - starting [357](#)
      - stopping [362](#)
    - function keys [245](#), [249](#)
    - governor
      - cancelled initialization [310](#), [316](#)
      - configuring message logging [303](#), [310](#)
      - level of detail, setting [303](#)
    - high-level qualifier for data sets (default) [34](#)
    - incorrect load modules [336](#)
    - initialization
      - Q.SYSTEM\_INI procedure [137](#)
      - level of detail [358](#)
      - level of detail, setting [141](#), [167](#), [172](#), [353](#)
      - message logging [340](#)
      - message number ranges and what they mean [353](#)
      - searching previously reported problems [365](#)
      - spill file [155](#), [349](#)
      - SQL codes, writing to DSQDEBUB [354](#)
      - SQLCA information, viewing [354](#)
      - starting a trace [356](#)
      - system errors [354](#)
      - TSO
        - diagnosis aids [364](#)
        - stored procedure interface [107](#), [119](#)
      - warning messages after starting QMF [344](#)
  - trademarks [408](#)
  - transaction
    - database
      - isolation levels [189](#)
    - routing requests with MRO and ISC [406](#)
  - transferring object ownership [208](#)
  - transient data queue
    - printing using QMF services [221](#)
    - routing output [213](#)
  - TRANSLATION column (Q.PROFILES) [172](#)
  - translations available in QMF, *See* National Language Feature (NLF)
  - troubleshooting
    - diagnostic aids [353](#)
    - dynamic statement cache search [352](#)
    - installation verification procedure (IVP)
      - DSQ21662 during import [77](#)
    - performance problems [347](#)
    - resource contention [351](#)
    - SQL codes
      - 551 [131](#)
    - storage requirements [349](#)
    - storage-related problems [349](#)
    - three-part name failures [160](#)
  - TSO (Time Sharing Option)
    - interface to governor [290](#)
    - interrupt facility [354](#)
    - storage requirements
      - initialization [37](#)
      - report operations [37](#)
      - setting dynamic allocation of storage [151](#)
      - setting fixed storage [151](#)
      - setting report storage as percentage [151](#)
      - setting variable storage amount [151](#)
    - stored procedure interface to QMF [107](#), [119](#)
    - virtual storage [150](#)
  - TSO user ID, using for QMF authentication [150](#)
  - TYPE column, QMF control tables [201](#)
  - TYPETERM specification [364](#)
- ## U
- U edit codes, forms
    - defined [258](#)
    - input area [254](#)
  - uncommitted read options [189](#)
  - uninstalling QMF
    - DB2 for iSeries and LUW databases [93](#)
    - DB2 for VSE and VM databases [95](#)
    - Db2 for z/OS databases [89](#)
  - unit of work, *See* remote unit of work or distributed unit of work
  - UNIX, *See* Linux, UNIX, and Windows platform
  - user
    - adding [181](#)
    - authentication [181](#)
    - authorization for objects [185](#)
    - objects [207](#)
  - user edit routines
    - handling different codes [257](#)
  - user-defined functions [401](#)
- ## V
- V edit codes, forms
    - defined [258](#)
    - input area [254](#)
  - Value Unit Edition, *See* VUE (Value Unit Edition)
  - values, variable [165](#)
  - VARBINARY data type
    - SQLTYPE, user edit routines [254](#)
    - width [153](#)
  - VARCHAR data type
    - SQLTYPE, user edit routines [254](#)
    - width [153](#)
  - VARGRAPHIC data type
    - SQLTYPE, user edit routines [254](#)
    - width [153](#)
  - variables
    - in synonym definitions [232](#)
    - passing values to initial procedure [165](#)
    - using &ALL [232](#)
  - VERB column (synonyms table) [229](#)
  - views
    - controlling access [185](#)
    - created during installation [13](#)
    - granting privileges for [200](#)
    - privilege to create [200](#)
    - privileges for queries [185](#)
    - Q.RESOURCE\_VIEW, governor exit [284](#)
    - screening tools [200](#)
    - underlying objects [200](#)
  - virtual storage, *See* storage
  - VM platform
    - authorities required [13](#)
    - database prerequisites [3](#)
    - packages available for governing [315](#)
    - QMF catalog column lengths [6](#)
    - remote unit of work configuration [5](#)
    - restrictions

- VM platform (*continued*)
  - restrictions (*continued*)
    - three-part names [8](#)
- VSAM data sets
  - used for indexes and table spaces [210](#)
- VSE platform
  - authorities required [13](#)
  - database prerequisites [3](#)
  - packages available for governing [315](#)
  - QMF catalog column lengths [6](#)
  - remote unit of work configuration [5](#)
  - restrictions
    - three-part names [8](#)
- VUE (Value Unit Edition)
  - license agreement [16](#), [132](#)
  - prerequisite database releases [3](#), [33](#)

## W

- warning messages [340](#), [344](#)
- WIDTH column (Q.PROFILES) [172](#)
- window panels
  - customized function key examples [249](#)
  - IDs [240](#)
- Windows, *See* Linux, UNIX, and Windows platform
- WLM address spaces required in QMF
  - stored procedure interface [107](#)
- WLM environment
  - updating for user-defined functions [401](#)

## X

- XCB fields of DXEXCBA control block
  - XCBAUTH [303](#)
  - XCBAUTHX [303](#)
  - XCBQRYP [303](#)
  - XCBQRYP2 [303](#)
  - XCBQRYPT [303](#)
- XML data type
  - extended storage for spill data [152](#), [153](#)
  - governing fetching of XML columns [315](#)
  - restrictions
    - exit routines for user edit codes [253](#)
    - multirow fetch [160](#)

## Z

- z/OS platform
  - authorities required [13](#)
  - database prerequisites [3](#)
  - DCB abend intercepts and spill file [349](#)
  - distributed unit of work configuration [8](#)
  - packages available for governing [315](#)
  - remote unit of work configuration [5](#)
  - upgrading Db2 but not QMF [30](#)







Product Number: 5697-QMF

GC27-8877

