DB2 Query Management Facility

Installing and Managing DB2 QMF for TSO/CICS

Version 8 Release 1

GC18-7444-00

DB2 Query Management Facility

Installing and Managing DB2 QMF for TSO/CICS

Version 8 Release 1

GC18-7444-00

Note

Before using this information and the product it supports, be sure to read the general information in Appendix F, "Notices," on page 385.

First Edition (January 2004)

This edition applies to IBM DB2 Query Management Facility for TSO/CICS (QMF for TSO/CICS) Version 8 Release 1, a feature of IBM DB2 Universal Database Server for z/OS (DB2 UDB for z/OS), Version 8 Release 1, 5625-DB2, and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this book

This book is intended to help database administrators and systems programmers install and manage the DB2 Query Management Facility (QMF) product under Time Sharing Option/Customer Information Control System (TSO/CICS) for z/OS™.

Who should read this book

This book is written for system programmers responsible for installing and managing DB2 QMF for use with IBM DB2 Universal Database for z/OS. It is also designed for network administrators responsible for installing and managing network applications. References to "Workstation Database Servers" in this book apply to:

- DB2 DataJoiner®
- DB2 Universal Database Version 7 and higher

What you should know before you begin

You should be familiar with the components that make up your specific environment.

z/OS

On z/OS, these components can include:

- Operating system z/OS
- Time Sharing Option (TSO) is an environment that supports DB2 QMF and its related products
- Interactive System Productivity Facility (ISPF), a dialog manager for DB2 QMF
- Customer Information Control System (CICS)®, is a general-purpose data communication and online transaction processing system. CICS/TS® provides the interface between DB2 QMF and z/OS.
- The base Graphical Data Display Manager (GDDM)® product is required regardless of printing requirements. GDDM makes it possible for QMF to display panels on the user's screen and create charts.
- DB2, the database manager for DB2 QMF

  DB2 also provides a number of utilities that can run in batch mode or through DB2I (the DB2 interactive facility) on z/OS.
- SMP/E (System Modification Program Extended)

- High-level assembly language (HLASM) is needed in order to modify or create a new governor exit routine. HLASM can also be used to create your own edit codes for QMF forms.
- PL/I is used to create your own edit codes in PL/I for QMF forms.
- VS COBOL II and COBOL are used to create your own edit codes in COBOL for QMF forms.
- REXX is used to create execs that install DB2 QMF.

Publications that discuss these products are listed in Appendix H, "Bibliography," on page 399.

How to use this book

The administration and customizing tasks in this book assume that QMF was installed according to the procedures described in this book. Most of the administration and customizing tasks are done using the DB2 QMF product itself. Before you begin the tasks in this book, ensure that the installation verification procedure (IVP) has been run. If not, run the IVP to ensure that QMF is properly installed and configured for your site's needs. The IVP is the final step of the QMF installation process.

Most of these tasks require that you have DB2 database administrator (DBA) authority. If the program installer follows the default procedure in this book, the user ID Q is defined for you during QMF installation; this user ID has DBA authority.

To keep the installation task as simple as possible, many of the full IBM product names and titles are shortened. Each product is referred to by its generic, rather than specific, name. For example, DB2 UDB for z/OS is DB2 and DB2 QMF is QMF.

How National Language Feature information is represented

DB2 QMF is available in several different languages, each of which is provided by a National Language Feature (NLF).

NLFs let users enter QMF commands, view help, and perform QMF tasks in languages other than English; they are installed as separate features of DB2 QMF. For more information about installing an NLF, see the NLF installation information in the appropriate operating system installation section in this book.

All tasks described in this book can be performed for the base DB2 QMF product (English language) and for any NLF. The procedures for both the base

and NLF sessions are the same; however, any special considerations for NLF users are preceded by the phrase: **If you are using an NLF**.

Some names of programs and phases shown in this book have a *n* in them, indicating that the name can vary. If you are using an NLF, replace any *n* symbols you see in this book with the one-character national language identifier (NLID) from Table 1 that matches the NLF that you installed. The table also shows the names by which QMF recognizes each language.

Table 1. NLIDs representing QMF base (English) and National Language Features (NLFs)

NLF	NLID	Name that QMF uses for this NLF
Brazilian Portuguese	P	PORTUGUES
Canadian French	C	FRANCAIS CANADIEN
Danish	Q	DANSK
English	E	ENGLISH
French	F	FRANCAIS
German	D	DEUTSCH
Italian	I	ITALIANO
Japanese	K	NIHONGO
Korean	H	HANGEUL
Spanish	S	ESPANOL
Swedish	V	SVENSKA
Swiss French	Y	FRANCAIS (SUISSE)
Swiss German	Z	DEUTSCH (SCHWEIZ)
Uppercase English	U	UPPERCASE

The uppercase feature (UCF) uses the English language, but converts all text to uppercase characters. The uppercase characters allow users working with Katakana terminals to use the product and get English online help and messages. Terminals equipped with Katakana support include IBM 3277, 3278, and 3279 terminals, as well as IBM 5550 Multistations.

How to order DB2 QMF books

To order hard copies, contact your IBM representative or visit the IBM Publications Center on the world wide web at: http://www.elink.ibmlink.ibm.com/applications/public/applications/publications/cgibin/pbi.cgi. Or, you can call 1-800-879-2755 in the United States or any of its territories.

How to send comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book, go to http://www.ibm.com/software/data/qmf/support.html, and click on Feedback.

Part 1. Installing QMF Version 8.1 for TSO/CICS

Chapter 1. Introducing QMF and the installation process

This chapter introduces the DB2 QMF host product, provides an overview of how QMF is installed, and describes how QMF connects to the different DB2 UDB databases.

Introducing QMF

QMF is a query and report writing program that lets users access databases and generate reports or charts based on the data they contain.

QMF runs under operating system z/OS, and primarily accesses data through DB2 UDB for z/OS. QMF works with both the Time-Sharing Option Extensions (TSO/E) and the online transaction manager under the control of the Customer Information Control System (CICS). CICS users can start QMF from within CICS and access data through the CICS/DB2 attachment.

In a host environment, QMF uses the IBM Graphical Data Display Manager (GDDM) to display panels. Display application panels can also be viewed with Interactive System Productivity Facility (ISPF). Figure 1 shows how these products relate to QMF in a host-only configuration.

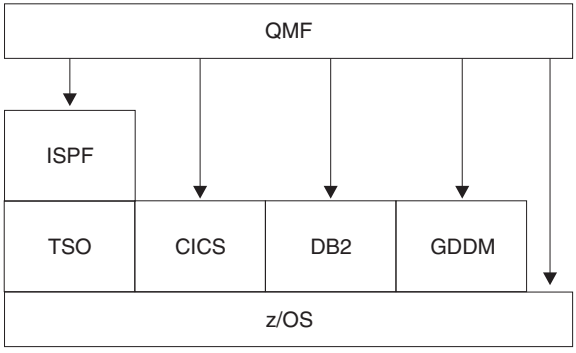


Figure 1. QMF in a host-only configuration.

QMF works with the following objects:

Data Information represented by alphanumeric characters contained in tables and formatted in reports.

Query Specifies the data you want and the action you want to perform.

Form Describes how retrieved data should be tailored into a report or chart.

Procedure
Contains one or more QMF commands that can be run as a group.

Profile
Contains information about how to process the user's session.

How QMF can access data in other databases

You can use QMF to connect to any of the DB2 UDB for z/OS, DB2 UDB for UNIX, Windows, and OS/2, DB2 Server for VSE or VM, DB2 UDB for iSeries, or DB2 UDB for Linux, UNIX, and Windows databases within a distributed network during QMF initialization or from within a QMF session. After successfully connecting to a location, you can access the data and QMF objects in that database in the same way you would access data and objects locally. For more information on the SQL CONNECT command, see *DB2 UDB for z/OS SQL Reference*.

QMF supports two methods of data access:
- Distributed Relational Database Architecture (DRDA) remote unit of work
- DB2 UDB for z/OS-to-DB2 UDB for z/OS distributed unit of work

DRDA is IBM's approach to distributed technology. Within DRDA there are different types of support such as remote unit of work, distributed unit of work, and distributed request. In the DRDA environment, QMF supports only remote unit of work.

DB2 UDB for z/OS-to-DB2 UDB for z/OS distributed unit of work allows you to access other DB2 subsystems using a communications method specific to DB2 UDB for z/OS. DB2 UDB for z/OS refers to this type of connection as system-directed access.

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.

Remote unit of work

This type of distributed access allows for reading or updating data at one remote location per unit of work.

DB2 UDB for z/OS Distributed Data Facility (DDF) adopted DRDA's data structure beginning with DB2 for OS/390 Version 2.3. With remote unit of work, DB2 UDB for z/OS can act as a server or requester (depending on the level of support from the partner system) for any remote database management system that implements DRDA.

If the startup program parameter DSQSDBNM or the QMF CONNECT command is used to specify a remote location to connect to, all subsequent QMF commands that access the database are directed to that location. (The CONNECT TO message appears on the QMF Home panel if DDF is installed.)

Figure 2 illustrates QMF with remote unit of work.

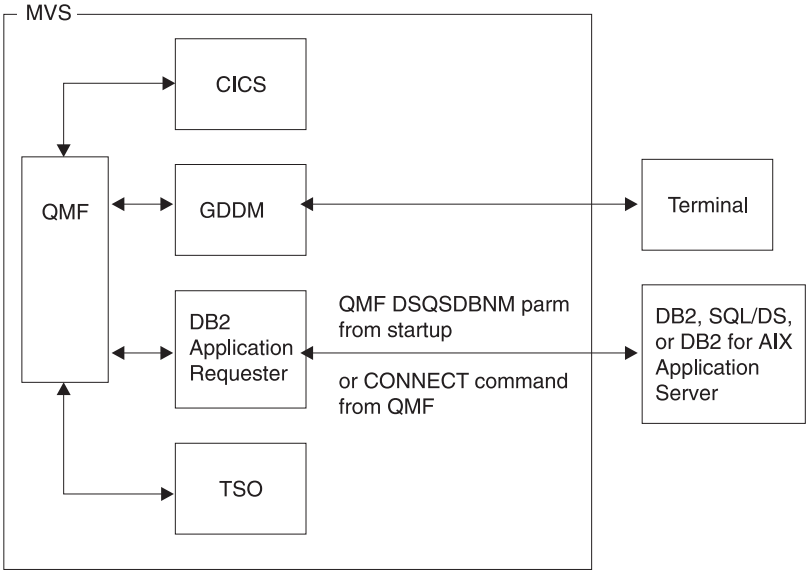


Figure 2. QMF using remote unit of work

DB2 UDB for z/OS distributed unit of work

This is an early version of distributed unit of work that was first introduced in DB2 for OS/390 Version 2.2. It allows access to other DB2 UDB for z/OS subsystems using a communications method that is private to DB2 UDB for z/OS. With this method you can connect to one location and run one query per unit of work. DB2 UDB for z/OS-to-DB2 UDB for z/OS distributed unit of work uses an alias or a three-part name to determine the location of the subsystem and to connect to it. The figure below shows a DB2 UDB for z/OS-to-DB2 UDB for z/OS access connection.

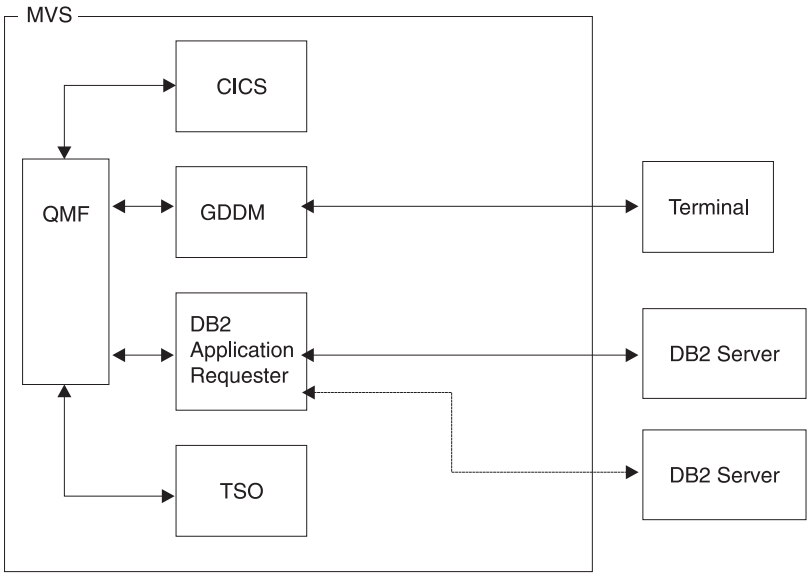


Figure 3. DB2 UDB for z/OS-to-DB2 UDB for z/OS connection

Overview of the database installation process

Installing DB2 QMF for TSO/CICS involves these object groups:

- QMF target and distribution libraries
- QMF application plan and packages
- QMF control tables, catalog views, and sample tables

QMF requirements for DB2 UDB for z/OS

QMF is a DB2 UDB for z/OS application program that uses standard interfaces to the database. QMF must be installed into at least one DB2 UDB for z/OS subsystem. Depending on the design of your data network, you may need to install QMF into additional DB2 UDB for z/OS subsystems.

Prerequisite DB2 UDB for z/OS knowledge

Because QMF is a DB2 UDB for z/OS application, you need to understand many of the same concepts as you would to perform a DB2 UDB for z/OS install. For example, you need to understand:

- CREATE, INSERT, and GRANT SQL statements

  You will use these statements during the QMF installation. These statements are described in more detail in *DB2 UDB for z/OS Reference*.
- The terms *application plan*, *DBRM*, *package*, and *bind*

These terms are described in the *DB2 for z/OS Application Programming and SQL Guide*.

- Databases, table spaces, tables, and views

  You need to understand the basic relationships among these terms. For information about these terms, see the *DB2 UDB for z/OS Administration Guide*.

- The DB2 UDB for z/OS security mechanism

  You need to understand what SYSADM and DBADM authority is and how to grant and revoke authority. You also need to understand the meaning of granting authority to PUBLIC. These topics are described in the *DB2 UDB for z/OS Administration Guide*.

- The ID of the DB2 subsystem where you plan to install QMF

  For information on subsystems IDs, see the *DB2 UDB for z/OS Administration Guide*.

Install QMF to access distributed data

You should be familiar with the terms:

- Application requester
- Application server
- Current location (current server)
- Distributed unit of work
- Local DB2 UDB for z/OS database
- Location name
- Remote unit of work

For definitions and more information about these terms, see *DB2 UDB for z/OS SQL Reference.*

DB2 UDB for z/OS objects created by QMF during installation

When QMF accesses a DB2 UDB for z/OS system, object types that were created for QMF during installation are available.

If you do not plan to install QMF into a distributed data environment, or if you plan to install QMF into a DB2 UDB for z/OS-to-DB2 UDB for z/OS distribution unit of work environment, you must install all of the following objects on each of the subsystems accessed by QMF:

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

For more information about these object types, see Appendix B, "QMF objects residing in DB2," on page 363.

Database authorization ID Q

Although the DB2 UDB for z/OS authorization ID of Q owns all control tables, sample tables, and catalog views in QMF, you do not need this authorization ID to install QMF. Without it, however, you need SYSADM authority.

If your authority (as installer) is revoked, the authorities granted during the installation process are also revoked, unless those privileges are also granted by some other authority.

Setting up QMF for remote unit of work

The simplest way to set up DB2 UDB for z/OS subsystems to use remote unit of work from QMF is to first run a full QMF installation, and then run a full database installation for each additional DB2 UDB for z/OS subsystem on the same z/OS system. After a DB2 subsystem (that supports remote unit of work) receives a full database installation, use that subsystem as either an application requester or application server for QMF. However, if you plan to use a particular DB2 UDB for z/OS subsystem as either an application requester or as an application server, install only those objects that are required.

Attention: The QMF CONNECT command works only when the instances of QMF being connected are of the same release.

Accessing data using remote unit of work

If you plan to use the DSQSDBNM startup program parameter or the QMF CONNECT command (both of these imply remote unit of work access) to connect to a remote location from QMF, you must first determine which DB2 UDB for z/OS subsystems function as application requesters and application servers for QMF.

- A subsystem that functions only as an application requester for QMF requires the QMF plan, one of the QMF packages (DSQIRDBR), and one of the QMF installation programs bound into that subsystem. These objects are created by the requester or full database installation option.
- A subsystem that functions as an application server for QMF requires the QMF packages, installation programs, control tables, catalog views, table space for SAVE DATA, and sample tables. Use the full or server database installation options to create these objects.
- A subsystem that functions as both an application requester and an application server requires the same objects as an application server alone. Use the full database installation option to create these objects.

Setting up QMF for DB2-to-DB2 distributed unit of work

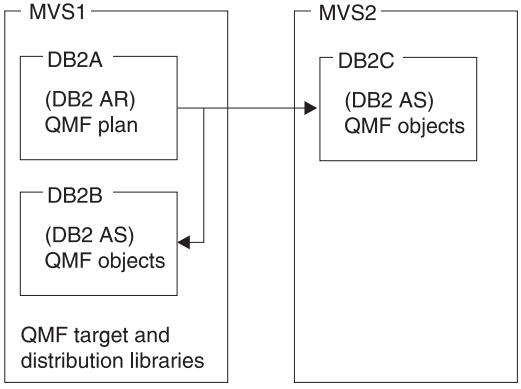
DB2 UDB for z/OS-to-DB2 UDB for z/OS distributed unit of work access to remote data is mostly transparent to QMF. Therefore, the install process you choose depends on whether you also plan to use remote unit of work. When using both remote unit of work and DB2 UDB for z/OS-to-DB2 UDB for z/OS distributed unit of work, the locations that you can access using three-part names are those that are accessible to the current server (if the current server is a DB2 location).

Example

The following example shows how to use the requester and server database installation options to install QMF in a remote unit of work environment:

Sample system configuration and requirements

- The z/OS operating system MVS1 has two DB2 UDB for z/OS Version 8.1 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 MVS2 has one DB2 UDB for z/OS Version 8.1 subsystem, DB2C. This system is the BATCH; DB2C is an application server, which is accessible to the TSO users on MVS1.
- QMF must be installed into DB2A as an application requester, and into DB2B and DB2C as application servers. Authorized users on DB2A can access data stored at DB2B and DB2C without logging on to different z/OS operating systems.



QMF objects are control tables, sample tables, views, application packages, and the application plan.

Installation sequence for the sample configuration:

1. On MVS1, install QMF target and distribution libraries.

2. On MVS1, use the requester database install option to install QMF into DB2A and customize the QMF run-time libraries.
3. On MVS1, use the server database install option to install QMF into DB2B. Use DB2A as the local DB2 and DB2B as the application server.
4. On MVS1, use the server database install option to install QMF into DB2C. Use DB2A as the local DB2 UDB for z/OS database and DB2B as the application server. You do not need to log on to MVS2, since the remote installation is run at MVS1.

Support for long names

Column, table, and other names up to 128 characters in length have been supported for some time in SQL. In Version 8.1, long names are supported in DB2 QMF for TSO/CICS. The following names are affected by this enhancement:

Table 2. Long names chart

Names	Compatibility mode for Version 7.2 or earlier	New Function mode (NFM)
Authorization ID	FIXED CHAR(8)	VARCHAR(128)
SQLID	FIXED CHAR(8)	VARCHAR(128)
Table, View, Synonym and Correlation	VARCHAR(18)	VARCHAR(128)
Column (see Note 1)	VARCHAR(18)	VARCHAR(30)
Location (see Note 2)	VARCHAR(16)	VARCHAR(16)

Note 1: Column names, currently described as VARCHAR(16) is extended to VARCHAR(128) in the DB2 catalog. However, Column names are limited to 30 bytes of Unicode in DB2 UDB for z/OS Version 8.1. DB2 QMF for TSO/CICS Version 8.1 supports a maximum length of 30 bytes.

Note 2: Location names, currently described as VARCHAR(16), is extended to VARCHAR(128) in the DB2 catalog. However, Location names are limited to 16 bytes of Unicode in DB2 UDB for z/OS Version 8.1. DB2 QMF for TSO/CICS Version 8.1 supports a maximum length of 16 bytes.

When DB2 UDB for z/OS Version 8.1 is running in NFM, long names in the database are supported. In all other modes, DB2 QMF for TSO/CICS Version 8.1 does not support long names in the database.

In addition to DB2 UDB for z/OS Version 8.1, the following name lengths are supported by DB2 QMF for TSO/CICS Version 8.1 when connecting to a remote server:

Table 3. Long name lengths supported when connecting to a remote server

	iSeries	UNIX and Windows	VSE & VM	z/OS
	Earlier than Version 2.0	Version 7.1	Version7.2 or earlier	Version 8.1 Compatibility mode or earlier
Schema name	10	8	8	8
Column name	10	18	18	18
Table name	18	18	18	18
	Version 2.0 or later	Version 7.1		V8.1 NFM or later
Schema name	10	8		128
Column name	30	30		30
Table name	128	128		128
		Version 7.1		
Schema name		30		

Compatibility mode versus New Function mode (NFM)

A QMF Compatibility mode installation is similar to an installation of a previous level of QMF. QMF owner names are limited to 8 characters and QMF object names are limited to 18 characters. A QMF Compatibility mode installation can co-exist in the same server with previous releases of QMF.

A QMF Version 8.1 New Function mode installation allows for QMF owner and object names as long as the database allows (maximum of 128 characters in length). **A QMF New Function mode installation cannot co-exist in the same server with a previous release of QMF.** The following servers will accept a QMF New Function mode installation:

- DB2 UDB for z/OS Version 8.1 server in NFM
- DB2 Universal Database Version 7.1 and higher
- DB2 for iSeries Version 5.1 and higher

A QMF NFM requester can connect to a QMF Compatibility mode server as long as the DB2 QMF Version 8.1 packages were bound to the server and the requester. Both the requester and the server must have the same release level package.

Users may migrate from QMF Compatibility mode to QMF New Function mode at any time. A QMF New Function mode installation can be done by either migrating from a QMF Compatibility mode installation, or into a server that does not contain a previous release of QMF. To migrate from a previous

release of QMF to QMF New Function mode, you must first migrate to QMF Compatibility mode, then migrate to QMF New Function mode.

DB2 QMF for TSO/CICS Version 8.1 migration overview

QMF Version 8.1 supports migration from QMF versions 3.3, 6.1, 7.1, and 7.2. A QMF migration installation will always produce a QMF Compatibility mode installation.

Road maps for the QMF installation

This section lists the QMF install types and paths for the various supported database servers. For more information about database objects that are created from an installation, see Appendix B, "QMF objects residing in DB2," on page 363

Initial installation or migration

- Read the program directory and complete the SMP/E installation as described
- Begin a DB2 UDB for z/OS database migration or new installation

You need to perform a new installation when:

- It is the initial installation of QMF into a DB2 UDB for z/OS database that contains no previous release of QMF
- It is the only installation of QMF into a DB2 UDB for z/OS database that contains no previous release of QMF
- More than one local DB2 UDB for z/OS subsystem must be accessed from QMF. Perform this type of installation for each additional local DB2 UDB for z/OS installation

See Chapter 3, "Configuring QMF as a DB2 application requester," on page 25 for more information on the install paths described above.

Perform a migration installation when the target DB2 UDB for z/OS database already contains QMF Version 3.3 or later. QMF Version 8.1 does not perform migration installations from QMF Version 3.2 or earlier. A QMF migration installation from version 3.3 or higher will always produce a QMF Compatibility mode installation.

See Chapter 3, "Configuring QMF as a DB2 application requester," on page 25 for more information on QMF migration installations.

Note: QMF Version 8.1 Compatibility mode should be thoroughly tested after migrating from QMF Version 3.3 or higher. Users should proceed from QMF Version 8.1 Compatibility mode to QMF Version 8.1 New Function mode with caution; once QMF Version 8.1 Compatibility mode is migrated to QMF New

Function Mode (NFM), no previous releases of QMF can be accessed in the same server. For more information on migrating to QMF NFM from QMF Compatibility mode, see Chapter 7, “Configuring QMF New Function mode for all server types,” on page 49.

Server database installations

If you plan to access data defined in a remote DB2 database that is accessible from your local DB2 UDB for z/OS subsystem, you must first perform a server database installation from your local subsystem. See Chapter 6, “Configuring remote servers for QMF Compatibility mode,” on page 41 for more information on Compatibility Mode installations and migration information; see Chapter 7, “Configuring QMF New Function mode for all server types,” on page 49 for information on QMF New Function mode installations into remote servers.

Chapter 2. Planning for installation

This chapter describes the hardware, program products, and direct access storage device (DASD) required to install and run QMF. It provides planning worksheets for easy reference during the install.

Hardware requirements

QMF runs on any processor supported by the operating system. QMF can access all the DASD devices supported by z/OS, and DB2 UDB for z/OS, and all terminals supported by GDDM.

If you plan to use the national language character set, you will need a workstation that supports the national language characters.

Prerequisite software

The following table lists program products and their minimum release levels that are necessary to support DB2 QMF Version 8.1. Later releases that are not available at the Version 8.1 announcement are not supported unless specifically stated otherwise.

Table 4. Prerequisite software for DB2 QMF for TSO/CICS Version 8.1

Software	Version	Part number
Any one of the following database software: DB2 Universal Database for z/OS DB2 Universal Database for OS/390 DB2 Universal Database for OS/390	Version 8 Release 1 Version 7 Release 1 Version 6 Release 1	5625-DB2 5675-DB2 5645-DB2
GDDM	Version 3 Release 2	5695-167
CICS/ESA **Note:** CICS/ESA is required only by CICS users.	Version 4 Release 1	5655-018

Either of the following program products and their minimum release levels are required to support optional functions for DB2 QMF for TSO/CICS Version 8.1. Later releases that are not available at the Version 8.1 announcement time are not supported unless specifically stated otherwise:

- DB2 Universal Database for OS/390 Version 7.1 with PTFs UQ57178, UQ60456, UQ60033, UQ66553 for Enhanced List Tables Command; part 5675-DB2

- DB2 Universal Database for OS/390 Version 6.1 with PTFs UQ57177, UQ60455, UQ60032 for Enhanced List Tables Command; part 5645-DB2

Planning your storage requirements

Make sure that there is enough storage to accommodate QMF programs and the QMF reports that users create. QMF storage requirements are as follows:

- QMF modules that can run in a 31-bit addressing mode require 4.5 MB.
- QMF modules that must run in a 24-bit addressing mode require 52 KB.
- The minimum storage amount for users to execute QMF queries and hold QMF report data is between 1.0 and 2.0 MB. Your specific requirements might be larger depending on the size of your report and the report formatting options used.

For example, if you run in a standard TSO environment with ISPF and GDDM, you will need approximately 8.0 MB of storage.

You can reduce the size of the region by placing ISPF and GDDM into the pageable link pack area (PLPA), which increases the common area accordingly.

TSO storage

You need 1.0 to 2.0 MB of storage space to run QMF. Additional storage space is required for other applications. For example, if you run QMF in a standard TSO environment with ISPF and GDDM, you need approximately 8.0 MB of storage space.

Most of the QMF modules are reentrant and can be loaded into EPLPA. The DSQCTOPX module must run in 24-bit mode below 16 MB; this module is also reentrant and can be loaded into PLPA.

CICS/ESA region

In CICS Version 3.1, dynamic storage area (DSA) can be allocated above and below 16 MB. The DSA above 16 MB is called extended DSA (EDSA). The DSA size is specified in the CICS system initialization table parameters DSASZE and EDSASZE. The CICS default value for EDSASZE, 1,536 KB, might be too small to support QMF users. Increase EDSASZE to the range of 16 to 50 MB, depending on the number of concurrent QMF users. Try using 16 MB plus 2 MB for each QMF concurrent user. For more information on this subject, see the appropriate CICS System Definition and Operations Guide.

Moving modules to enhance performance

The library QMF810.SDSQLOAD contains the load modules for the QMF product. Table 5 shows the modules you can move into link pack area libraries to enhance performance.

Table 5. Modules that can reside in the PLPA or EPLPA

Module	Description
DSQQMFE DSQQMF DSQCSUB DSQCTOPX DSQCCI DSQCCISW DSQCBST DSQCELTT DSQCEBLT DSQCIX	QMF uses the modules in this set when you invoke QMF. DSQCTOPX and DSQCCI can be placed only in the PLPA.
DSQUEDIT DSQUXIA DSQUXIC DSQUXILE DSQUXIP	These modules are related to the user EDIT routines. Unless you expect heavy use, do not move them into the link pack area.
DSQCIB COBOL DSQCICX C/370 DSQCIA assembler DSQCIFE FORTRAN DSQCIF FORTRAN DSQCIPX PL/1 DSQCIPL PL/1 DSQCIR RPG DSQCIX REXX	The QMF callable interface uses the modules in this set, which 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.
DSQUEGV3	This is a governor module.

Table 6 describes the modules that cannot be placed in the PLPA or EPLPA.

Table 6. TSO Modules that can't reside in the PLPA or EPLPA

Module	Description
DSQCI	QMF uses this module when QMF is invoked.

Table 6. TSO Modules that can't reside in the PLPA or EPLPA (continued)

Module	Description
DSQUEGV1	This module is a governor routine.
DSQCMAPB DSQ0BINS DSQ0BSQL DSQCTO80 DSQCFR80	These modules are QMF installation and service updates.

In CICS

QMF runs as a conversational transaction in CICS where there are multiple users of QMF in the same CICS address space. Each user that runs a QMF transaction requires at least 1.0 MB of storage from the CICS region. You can allocate all but 24 KB to storage above 16 MB. You can place a single copy of the QMF module, up to 2.7 MB, in the EPLPA or within the CICS region above 16 MB, and you can place 52 KB in PLPA or within the CICS region below 16 MB.

Estimating storage

System Modification Program Extended (SMP/E) is the basic tool that you use to install QMF. With SMP/E, you install into two types of libraries:

- Target libraries, which contain the executable code making up the running system
- Distribution libraries, which contain the master copy of all the system elements

Refer to the QMF Program directory for estimating the size of all SMP/E and QMF Target and Distribution library data sets.

Read the Program Directory and apply service

Before beginning the installation process, read the QMF Program Directory for supplementary data. Because the Program Directory is updated between releases of QMF, it contains useful information including descriptions of program temporary fixes (PTFs) and authorized program analysis reports (APARs), as well as modifications to this book.

Ensure that the service level of your system is current. Call your IBM Software Service Support, or use IBMLink (ServiceLink) in the United States or EMEA DIAL in Europe, to request the latest PTFs for QMF and its prerequisite products. Additionally, request QMF's preventative service planning (PSP) bucket. The QMF PSP bucket subset and upgrade name can be found in the QMF Program Directory. The PSP bucket contains general hints,

HIPER APARs, and documentation changes. Subscribers who have access to either Information/Access or ServiceLink can download the information from the Internet.

Planning for QMF under CICS

You need to complete installing, tailoring, and testing CICS and GDDM before you install QMF.

Tailoring CICS for QMF

Because QMF is a large conversational transaction, QMF processing takes longer than the average CICS transaction. You might want to isolate QMF transaction processing in a CICS region dedicated to QMF transactions.

Depending on the amount of storage available below 16 MB, there is an upper limit on the number of users that can run QMF in the same CICS region. To support additional QMF users, use multiple CICS regions and the Multiple Region Option.

You might want to route the QMF transaction from one CICS system (for example, Terminal Owning Region) to the CICS system designated to process QMF transactions (for example, Application Owning Region). If you do, use either multiple transaction IDs or dynamic transaction routing. Both methods are described in the *CICS Intercommunication Guide*.

Tailoring GDDM for QMF

During QMF installation, QMF modifies GDDM's ADMF file. Additionally, you must define GDDM resources, such as programs and transactions, to CICS. For details on how to install and tailor GDDM, see the *GDDM Installation and System Management* guide.

Changing GDDM default parameters

If you are using GDDM Version 2.3, ensure that the IOSYNCH parameter in the ADMADFC external defaults module is set to YES.

Run the installation verification procedure (IVP) for GDDM

Run the IVP for GDDM. The IVP minimizes QMF installation problems and ensures that you are installing QMF onto a clean system.

QMF with DB2 for AIX

Customizing QMF to work with a DB2 for AIX server require some changes both on the host and the server.

QMF uses the distributed data facility (DDF) to access distributed data that resides in a DB2 UDB for AIX database. The DDF of DB2 UDB for z/OS is a VTAM application that uses LU 6.2 communications protocols to communicate

with other database management systems or applications that support Distributed Relational Database Architecture (DRDA). Information about connecting distributed database systems for access to data from QMF for TSO/CICS is in the *DB2 UDB for z/OS Administration Guide*.

From DB2 UDB for z/OS, the communications database (CDB) tables are used to control access between remote database management systems. If you plan to use DB2 UDB for z/OS as a server only, you do not need to populate the CDB; default values are used. However, if you intend to request data from remote databases, you must update the CDB tables. These topics are described in the *DB2 UDB for z/OS Installation Guide*.

At the DB2 UDB for AIX server, you must issue a CREATE DATABASE command before installing QMF into that database. Verify that APPC communications are defined and operational between the DB2 UDB for z/OS for z/OS DRDA application requester and the DB2 for AIX DRDA application server.

For more information about the installation of QMF objects into DB2 UDB for AIX from DB2 UDB for z/OS, and about the prerequisites for the installation, see Chapter 6, “Configuring remote servers for QMF Compatibility mode,” on page 41.

Complete the worksheets

Table 7, displays the required information you need to provide values for during the QMF installation. Use them as worksheets.

Table 7. QMF installation required information (Version 8.1 Worksheet part 1

REQUIRED INFORMATION	VALUE
Location name	
Target Library Prefix (Default = QMF810)	
Local DB2 UDB for z/OS Subsystem ID (Default=DSN)	
Local DB2 UDB for z/OS Exit Library (Default=DSN810.SDSNEXIT)	
Local DB2 UDB for z/OS Load Library (Default=DSN810.SDSNLOAD)	
QMF application plan ID (Default=QMF810)	

REQUIRED INFORMATION	VALUE
DB2 UDB for z/OS user catalog (ICF) (Default=DSNC8101.USER.CATALOG)	
QMF table space catalog alias (Default=QMFDSN)	
QMF tables volume	
DB2 UDB for z/OS default punctuation	, (comma) or . (period)
Previous QMF level (migration installs only)	V3R3, V6R1, V7R1, V7R2, or NONE

REQUIRED INFORMATION	PRIMARY	SECONDARY
The following information applies to DB2 UDB for z/OS servers that contain no previous levels of QMF. QMF recommends the following values, but they may be overridden at installation time.		
QMF control table table space Sizes: (in 1K units) Table Space name Default size (primary, secondary) - Q.OBJECT_DIRECTORY (see Note) (200,20) - Q.OBJECT_REMARKS " (200,20) - Q.OBJECT_DATA " (5000,200) - Q.PROFILES " (100,20) - Q.ERROR_LOG " (100,20) - Q.COMMAND_SYNONYMS " (100,20) - Q.RESOURCE_TABLE " (100,20) - Q.DSQ_RESERVED (100,20) - SAVE DATA (Optional) " (100,20)	_____ _____ _____ _____ _____ _____ _____ _____ _____	_____ _____ _____ _____ _____ _____ _____ _____ _____
Table Index Sizes: (in 1K units) Table Index name Default size (primary, secondary) - Q.OBJECT_DIRECTORYX (see Note) (200,20) - Q.OBJECT_REMARKSX " (200,20) - Q.OBJECT_OBJDATAX " (200,20) - Q.PROFILEX (200,20) - Q.COMMAND_SYNONYMSX " (100,20)	_____ _____ _____ _____ _____	_____ _____ _____ _____ _____
Determine the following, if applicable:		
Do you want SAVE DATA table space created?	yes or no	

Note: Control tables and indexes are provided only on the initial installation of QMF.

QMF installation job information

New in DB2 QMF for TSO/CICS Version 8.1: The DSQ1EINS installation CLIST and panels have been removed from the product. Submitting batch installation jobs is the only way to install QMF. The jobs DSQ1EMAP, DSQ1CHRT, DSQ1EJVE, and DSQ1EJVC have also been removed from the QMF installation. The running of the installation VSAM panel job DSQ1EPNL and its information has been moved to Section 6.2 in the Program Directory.

In the following QMF installation chapters, all job names can be found as members in the QMF810.SDSQSAPE data set. All jobs are liberally commented and should be consulted for tailoring and submission details.

All QMF database installation defaults that are available for override can be found in QMF810.SDSQEXCE(DSQ1DEFS). The DSQ1DEFS REXX exec also references every job that uses each of the defaults. These defaults can also be found in Table 7 on page 20. Some examples of variables contained in DSQ1DEFS are QMF control table space and indexspace primary and secondary quantity values, and the VCAT and VOLUMES parameters for the QMF stogroup. Not all QMF installation jobs use every value in DSQ1DEFS. The jobs clearly describe each DSQ1DEFS value that is referenced; if a DSQ1DEFS value is not referenced in a job, it is ignored. **No variables should ever be deleted from DSQ1DEFS.**

If the QMF installer wants to modify any DSQ1DEFS values, he should first create a copy. The copy can then be modified, and the QMF installation job DSQDEFS ddcard should be modified to reference the copy.

QMF uses this new DSQ1DEFS exec in most database installation jobs. Two examples are provided to show the flexibility of this exec and the new installation process it provides:

In example 1, a user is running the installation job DSQ1TBLJ. He is satisfied with the QMF default values specified in DSQ1DEFs for all values, and needs to specify overrides for SSID, LOCATION, and VCATNAME. This example illustrates the modified SYSTSIN statement:

```
//DSQ1TBLJ JOBcard
//*comments
//DSQ1TBLJ PROC RGN='2048K',
//     QMFTPRE='QMF810',
//     DB2EXIT='DSN810.SDSNEXIT',
//     DB2LOAD='DSN810.SDSNLOAD'
//*-------------------------------
```

```
//* CREATE AND LOAD QMF DATABASE AND CONTROL TABLES FOR QMF --
//* COMPATIBILITY MODE.      ---
//* ------------------------------
//STEP1 EXEC PGM=IKJEFT01,REGION=&RGN
//STEPLIB DD DSN=&MFTPRE..SDSQLOAD,DISP=SHR
//    DD DSN=&DB2EXIT.,DISP=SHR
//    DD DSN=&DB2LOAD.,DISP=SHR
//SYSTPRT DD SYSTOUT=*,DCB=BLKSIZE=121
//SYSTERM 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(DSQ1VSTG),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLB),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLI),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLU),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLE),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLN),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLG),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLK),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBDC),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLC),DISP=SHR
//   PEND
//DSQTBL EXEC DSQ1TBLJ
//*============================================
//* Tailor below:
//*============================================
//STEP1.SYSTSIN DD*
%DSQ1INST QMFBSQL SSID(DB2L) LOCATION(MVS1DB2L) +
    VCATNAME(DB2LDSN) VOLUMES('*')
```

In example 2, a user is running the installation job DSQ1TBLJ. He is satisfied with the QMF default values specified in DSQ1DEFs for all values, and needs to specify overrides for SSID, LOCATION, and VCATNAME. He wants to do the overrides from the DSQ1DEFS ddcard. Note that the copy can be made to any member name.

Copy QMF810.SDSQEXCE(DSQ1DEFS) to QMF810.SESQEXCE(DB2LDEFS). Edit QMF810.SDSQEXCE(DB2LDEFS) to change values:

```
SSID = "DB2L"
LOCATION = 'MVS1DB2L"
VCATNAME = ""DB2LDSN"
```

Modify the DSQ1TBLJ DSQDEFS ddcard to point to the copied and modified member DB2LDEFS, and remove the override options from SYSTSIN. Make sure to leave the QMFBSQL value after the DSQ1INST call:

```
//DSQ1TBLJ JOBcard
//*comments
//DSQ1TBLJ PROC RGN='2048K',
//     QMFTPRE='QMF810',
//     DB2EXIT='DSN810.SDSNEXIT',
```

```
//     DB2LOAD='DSN810.SDSNLOAD'
//*--------------------------------------
//* CREATE AND LOAD QMF DATABASE AND CONTROL TABLES FOR QMF ---
//* COMPATIBILITY MODE.          ---
//*--------------------------------------
//STEP1 EXEC PGM=IKJEFT01,REGION=&RGN
//STEPLIB DD DSN=&QMFTPRE..SDSQLOAD,DISP=SHR
//    DD DSN=&DB2EXT.,DISP=SHR
//    DD DSN=&DB2LOAD.,DISP=SHR
//SYSTSPRT DD SYSOUT=*,DCB=BLKSIZE=121
//SYSTERM DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSEXEC DD DSN=&QMFTPRE..SDSQEXCE,DISP=SHR
//DSQDEFS DD DSN=&QMFTPRE..SDSQEXCE(DB2LDEFS),DISP=SHR <---Modified
//DSQINDD DD DSN=&QMFTPRE..SDSQSAPE(DSQ1VSTG),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLB),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLI),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLU),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLE),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLN),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLG),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBLK),DISP=SHR
//   DD DSN=&QMFTPRE..SDSQSAPE(DSQ1TBDC),DISP=SHR
//   DD DSN+&QMFTPRE..SDSQSAPE(DSQ1TBLC),DISP=SHR
//   PEND
//DSQTBL EXEC DSSQ1TBLJ
//*=============================================
//* Tailor below:
//*=============================================
//STEP1.SYSTSIN DD *
%DSQ1INST QMFBSQL      <-- Modified (Be sure to leave the order
  specified afterDSQ1INST).
```

Once DB2LDEFS is modified for the DB2L subsystem, it can be used in all QMF installation jobs. For example, the user might also use DB2LDEFS in DSQ1BVW, DSQ1STGJ, and DSQ1EIVS. A user could modify a DSQ1DEFS per server install to use in all QMF installation jobs for that server.

Chapter 3. Configuring QMF as a DB2 application requester

One of the following installation types should be done for each DB2 UDB for z/OS local application requester:

- Install QMF Compatibility mode
- Install QMF New Function mode
- Migrate to QMF Compatibility mode

The following sections reference job names in their task tables. These jobs, which reside as members in the QMF810.SDSQSAPE data set, are heavily commented and must be tailored and run in their specified order to successful completion.

Installing QMF Compatibility mode

This series of steps will install and prepare QMF in Compatibility mode in a DB2 for OS/390 Version 6.1 (or later) server that contains *no* previous release of QMF.

Table 8. Job sequence for new QMF Compatibility mode install into DB2 UDB for z/OS server

Job name	Purpose
DSQ1TBAJ	Optional: Creates QMF VCAT name
DSQ1BSQL	Binds QMF installation packages and plan to a target server
DSQ1TBLJ	Creates QMF control tables
DSQ1BVW	Creates QMF Version 8.1 views
DSQ1BPKG	Binds QMF packages
DSQ1BINR	Binds QMF application plan (DB2 UDB for z/OS local installations only)
DSQ1STGJ	Creates the SAVE DATA table space for the QMF IVP
DSQ1EIVS	Creates QMF sample tables

Installing QMF New Function mode

The following tasks in the table below will install QMF New Function mode into a DB2 Version 8.1 NFM server that contains *no* previous release of QMF.

Table 9. Installing QMF New Function mode

Job	Purpose
DSQ1TBAJ	Optional- Creates QMF VCAT name
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1BLNI	Creates QMF NFM control tables
DSQ1BVW	Creates QMF views
DSQ1BPKG	Binds QMF application packages
DSQ1BINR	Binds the QMF application plan
DSQ1STGJ	Creates the SAVE DATA table space for the QMF IVP
DSQ1EIVS	Optional- Creates the QMF sample table

Migrating to QMF Compatibility mode

The steps in the table below will migrate a QMF Version 3.3, 6.1, 7.1, or 7.2 installation in a DB2 for OS/390 Version 6 or later server to a QMF Version 8.1 Compatibility mode installation. Previous releases of QMF can coexist with QMF Compatibility mode.

Table 10. Job sequence for migration from QMF Version 3.3 or higher to QMF Compatibility mode

Job	Purpose
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1BVW	Creates QMF views
DSQ1BPKG	Binds QMF application packages
DSQ1BINR	Binds the QMF application plan

At this point, you are ready to tailor QMF for TSO or CICS. See Chapter 4, “Tailoring QMF for TSO,” on page 27 for TSO and Chapter 5, “Tailoring QMF for CICS,” on page 35 for CICS.

Chapter 4. Tailoring QMF for TSO

This chapter describes the tailoring of QMF for TSO. It includes the following steps:

- Create a TSO logon procedure
- Start QMF
- Set up QMF batch job to run batch IVP (optional)

Create a TSO logon procedure

DSQ1EINV is an IBM-supplied sample TSO procedure.

Starting QMF in TSO

ISPF users can start QMF with the ISPF SELECT service and the ISPSTART commands. Without ISPF, users can use the DSQQMFE module. For more information on ISPF dialogs, see *Interactive System Productivity Facility for OS/390 Dialog Management Services and Examples*.

As the QMF installer, you must have a TSO logon procedure. When you log on to TSO as installer and start the Terminal Monitor Program (TMP), it invokes the TSO logon procedure.

The TMP is the principal interface between user and terminal during the user's TSO sessions. Your installation might be using either its own TMP or the standard one supplied by IBM. If the TMP is not the standard one, some of the following information might not apply.

Besides invoking the TMP, a logon procedure allocates resources for its users at the start of a TSO session. QMF users need more resources than the minimum set that all TSO users require. By using a logon procedure, you ensure that you are providing these additional resources to establish an adequate TSO environment.

The TSO logon procedure starts when a user logs on to TSO. After the procedure runs, you have the option of also running a logon CLIST.

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 may be required.

Some of the resources that are allocated in the logon procedure can also be allocated in a CLIST that invokes QMF.

Preparing the TSO logon procedure

1. Edit QMF810.SDSQSAPE(DSQ1EINV).
2. Locate the region parameter and ensure that it meets the minimum storage requirements as described in “Planning your storage requirements” on page 16.

   ```
   //DSQ1EINV  EXEC PGM=IKJEFT01,TIME=1440,DYNAMNBR=30,REGION=4096K
   ```
3. Review the program load libraries.
   a. Determine whether you want to allocate the program modules through the STEPLIB statement or through a CLIST. Add the QMF user exit library QMF810.SDSQEXIT to the STEPLIB concatenation if needed. This only needs to be done if any exits reside in QMF810.SDSQEXIT.

      The sample includes the load libraries for ISPF, ISPF-PDF, QMF, DB2 UDB for z/OS, and GDDM. Not all of these libraries need to appear in 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.

   b. Tailor for ISPF, if appropriate.

      If you are running with ISPF, you can make the STEPLIB allocation with the ISPF ISPLLIB DD statement.

   c. Determine whether you want to run concurrent versions of QMF on the same DB2 UDB for z/OS subsystem.

      If you plan to run concurrent versions of QMF with different plan IDs on the same DB2 UDB for z/OS database, you cannot use the same QMF load library in the same procedure. The following list indicates the load module library names for QMF versions.

      QMF Version
      Load Module Library Name
      Version 8 Release 1.0
      QMF810.SDSQLOAD
      Version 7 Release 2.0
      QMF720.SDSQLOAD
      Version 7 Release 1.0
      QMF710.SDSQLOAD
      Version 6
      QMF610.SDSQLOAD
      Version 3 Release 3.0
      QMF330.DSQLOAD

```
//*********************************************************
//*        PROGRAM LOAD LIBRARIES                         *
//*********************************************************
//STEPLIB  DD  DSN=QMF810.SDSQEXIT,DISP=SHR      * QMF MODULES *
//         DD  DSN=QMF810.SDSQLOAD,DISP=SHR      * QMF MODULES *
//         DD  DSN=ISR.V4R1M0.ISRLOAD,DISP=SHR   * PDF MODULES * Opt. for
//                                                    non-ISPF users
//         DD  DSN=ISP.V4R1M0.ISPLOAD,DISP=SHR   * ISPF MODULES * Opt. for
```

```
//                                                  non-ISPF users
//         DD  DSN=DSN810 .SDSNEXIT,DISP=SHR     * DB2 MODULES  *
//         DD  DSN=DSN810 .SDSNLOAD,DISP=SHR     * DB2 MODULES  *
//         DD  DSN=GDDM230.SADMMOD,DISP=SHR      * GDDM MODULES *
```

4. Allocate SDSQEXCE to either SYSEXEC or SYSPROC.

   Use the DDNAME established by your installation for the TSO search order for execs. 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 EXECs, allocate SDSQEXCE to both SYSEXEC and SYSPROC.

```
//*********************************************************
//*        DATA SETS USED BY TSO                          *
//*********************************************************
//SYSPROC  DD  DSN=SYS2.CLIST,DISP=SHR            * CLIST Library
//         DD  DSN=QMF810.SDSQCLTE,DISP=SHR
//SYSEXEC  DD  DSN=SYS2.EXEC,DISP=SHR
//         DD  DSN=QMF810.SDSQEXCE,DISP=SHR
//SYSHELP  DD  DSN=SYS1.HELP,DISP=SHR
//EDT      DD  DSN=&EDIT,UNIT=SYSDA,SPACE=(1688,(40,12))
```

5. Tailor ISPF libraries, if appropriate.

   ISPF libraries are optional. If you use ISPF-related functions, allocate these libraries.

```
//*********************************************************
//*        DATA SETS USED BY ISPF                         *
//*********************************************************
//ISPPLIB  DD  DSN=QMF810.SDSQPLBE,DISP=SHR    * Panel libraries
//         DD  DSN=ISR.V4R1M0.ISRPLIB,DISP=SHR
//         DD  DSN=ISP.V4R1M0.ISPPLIB,DISP=SHR
//ISPMLIB  DD  DSN=QMF810.SDSQMLBE,DISP=SHR    * Message Libraries
//         DD  DSN=ISR.V4R1M0.ISRMLIB,DISP=SHR
//         DD  DSN=ISP.V4R1M0.ISPMLIB,DISP=SHR
//ISPSLIB  DD  DSN=QMF810.SDSQSLBE,DISP=SHR    * ISPF Skeleton Libraries
//         DD  DSN=ISR.V4R1M0.ISRSLIB,DISP=SHR
//         DD  DSN=ISP.V4R1M0.ISPSLIB,DISP=SHR
//ISPTLIB  DD  DSN=ISR.V4R1M0.ISRTLIB,DISP=SHR * Table Input Libraries
//         DD  DSN=ISP.V4R1M0.ISPTLIB,DISP=SHR
//ISPPROF  DD  UNIT=SYSDA,SPACE=(TRK,(9,1,4)), * User's ISPF Profile Library
//         DCB=(LRECL=80,BLKSIZE=8800,RECFM=FB,DSORG=PO)
```

6. Verify GDDM data sets.

   These are allocated to ddnames beginning with ADM.

   a. Ensure ADMGGMAP and the ADMGGMAP library are allocated properly.

   b. Allocate separate libraries for users who want to save their own chart forms. Create the new library with a DD statement like this:

```
//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.

1) Locate the entry for DSQUCFRM in DSQ1EINV.
2) Replace aaaaaaa with the name of the user's library.
3) Duplicate and customize this entry for each user library.

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 the following statements:

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

```
//*********************************************************
//*        QMF/GDDM DATA SETS                             *
//*********************************************************
//ADMGGMAP  DD  DSN=QMF810.SDSQMAPE,DISP=SHR   * GDDM Map Group
//ADMCFORM  DD  DSN=QMF810.SDSQCHRT,DISP=SHR  * QMF-Supplied Chart Forms
//DSQUCFRM  DD  DSN=aaaaaaa,DISP=SHR          * Saves User-Defined ICUFORMS
//ADMCDATA  DD  DSN=xxxx,DISP=SHR
//ADMGDF   DD DSN=xxxx,DISP=SHR
//ADMSYMBL DD DSN=xxxx,DISP+SHR
```

7. Tailor for QMF preferences.

   Data sets DSQDEBUG, DSQUDUMP, and SYSUDUMP all currently default to a printer. You can tailor the definition to send the information instead to a data set.

   DSQUDUMP, DSQDEBUG, and DSQPRINT all require a DCB parameter. For DSQPRINT, add 1 to LRECL for the print control character.

```
//*********************************************************
//*        DATA SETS USED BY QMF                          *
//*********************************************************
//DSQPNLE   DD  DSN=QMF810DSQPNLE,DISP=SHR                      * Panel Definition File
//DSQPRINT  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)    * Print Output
//DSQDEBUG  DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)    * Trace Output
//DSQEDIT   DD  UNIT=SYSVIO,DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029), * Edit Transfer File
//  DISP=NEW,SPACE=(CYL,(1,1))
//DSQUDUMP  DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)    * Snap Dump Output
//SYSUDUMP  DD  SYSOUT=A
//DSQSPILL  DD  DSN=&&SPILL,DISP=(NEW,DELETE),                   * User's Spill File
//  UNIT=SYSVIO,SPACE=(CYL,(1,1),RLSE),
//  DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
```

Start QMF

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

Starting QMF with ISPF

1. Start QMF from an application program using the callable interface, or issue the ISPSTART command with or without parameters. The following examples show how to use ISPSTART to override the default values for the database subsystem name (DSN) and the plan ID (QMF810).
   - With parameters:

     Choose the appropriate command for your type of install. If you are installing QMF into another DB2 UDB for z/OS subsystem, the value for `ssid` must be changed to your subsystem ID value.
     - Full installs:

       ```
       ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE)
        PARM(DSQSSUBS=ssid,DSQSPLAN=planid,...)
       ```
     - Server installs:

       ```
       ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE)
       ```
     - Requester installs:

       ```
       ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSSUBS=ssid,
       DSQSPLAN=planid,DSQSDBNM=<location>,...)
       ```

     The QMF Home panel is displayed. After the QMF session ends, you are returned to TSO READY mode.

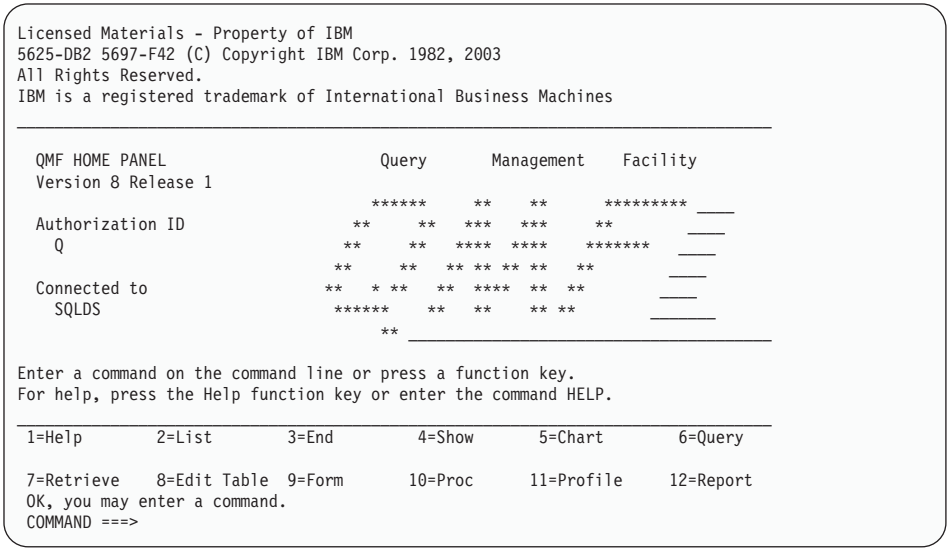


Figure 4. QMF Home Panel

   - Without parameters:

     ```
     ISPSTART
     ```

You should now see the ISPF Master Application menu. From here, you can select QMF. After the QMF session ends, the ISPF Master Applications menu returns. The following section explains how to customize the ISPF selection menus to include QMF.

Customizing the ISPF selection menus

ISPF supplies a master application menu as part of its installation process. You can invoke QMF from the ISPF Master Application menu or from any other selection menu that you want to use. Figure 5 on page 33 shows an example of how to code the ISPF Master Application menu to include QMF. The line for QMF is option 2.

You can change the program parameters you pass from TSO to QMF by using the QMF callable interface REXX procedure QMF810.SDSQEXCE(DSQSCMDE). Another method of passing program parameters is through the ISPF service call that QMF uses.

```
%------------------------  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(DSQQMFE) NEWAPPL(DSQE)'
           /*                             */
           /* ADD OTHER APPLICATIONS HERE */
           /*                             */
                P,'PANEL(ISPOPT)'
                X,'EXIT'
              ' ',' '
                *,'?' )
)END
```

Figure 5. QMF Dialog on ISPF Master Application Menu

Starting QMF in TSO

To reach QMF, you can use an application program and the callable interface. Enter the following statement:

```
DSQQMFE DSQSSUBS=dbname,DSQSPLAN=planid,...
```

where (...) represents additional parameter values being passed to QMF.

You can also start QMF with the following TSO call commands:

- For full or server installs:

  ```
  CALL 'QMF810.SDSQLOAD(DSQQMFE)' 'DSQSSUBS=dbname,DSQSPLAN=planid,...'
  ```

- For requester installs:

```
CALL 'QMF810.SDSQLOAD(DSQQMFE)' 'DSQSSUBS=dbname,DSQSPLAN=planid,
      DSQSDBNM=<location>...'
```

Set up a QMF batch job to run batch IVP (optional)

In this step you set up a batch job for the batch-mode IVP. If you want to run this test, you must wait until "Optional: Run the batch-mode IVP" on page 57. If you run the test earlier, the test will fail because the procedure Q.DSQ1EBAT is not yet available.

To create a batch job:

1. Make a copy of the sample logon procedure (DSQ1EINV).
2. Add a JOB statement.

   If you are working in a RACF environment, make the value of the USER parameter the logon ID of the installer. For example, if the installer is JONES, the job statement might look like this:

   ```
   //BATCH JOB USER=JONES,PASSWORD=password
   ```

   where *password* is JONES' password.
3. Delete the SYSTERM and SYSIN DD statements.
4. Add the following statements to the end of the logon procedure:

   ```
   //SYSTSPRT DD SYSOUT=A
   //SYSTSIN DD *
         PROFILE PREFIX(JONES)
         ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(M=B,I=Q.DSQ1EBAT,S=ssid)
   /*
   ```

   The first control card within the second JCL statement is optional. Use it if your installation does not have RACF. Replace JONES with the logon ID of whoever is running the step.

   The second control card within the second JCL statement invokes QMF in batch mode (DSQSMODE=B). Replace `ssid` with the subsystem ID of the database subsystem into which you installed QMF. If you do not specify a subsystem ID, the default, `DSN`, will be used. When invoked in this way, QMF invokes the procedure Q.DSQ1EBAT. After it invokes the procedure, control returns to TSO, which terminates the job because it finds no more TSO statements in SYSTSIN.

Proceed to Chapter 8, "Testing your QMF installation," on page 51.

Chapter 5. Tailoring QMF for CICS

This chapter describes the steps required to tailor QMF for CICS.

Before performing the tailoring process for QMF in CICS, you must install and tailor DB2 UDB for z/OS, and GDDM for CICS. For more details, see *GDDM Installation and System Management* and *DB2 UDB for z/OS Administration Guide*.

Describe QMF to DB2 in CICS

1. Ensure that you install the DB2 UDB for z/OS-to-CICS connection and the DB2 UDB for z/OS attachment facility for CICS.

   QMF uses the CICS/DB2 attachment facility to access DB2 UDB for z/OS data in the CICS environment. For more information on setting up DB2 UDB for z/OS in a CICS environment, see the *CICS DB2 Guide.*
2. Verify that the CICS RDO definitions exist for DB2CONN. The QMF-specific DB2ENTRY definition will be created when the DSQ1ECDN job is run.

   Users invoking a QMF transaction operate under the authorization of the associated RCT entry.

   All QMF programs are bound during installation; it is unnecessary to separately bind for CICS.

Define and load QMF/GDDM data sets

This step defines and loads a number of data sets.

- DSQ1EADM loads QMF/GDDM map sets to the GDDM ADMF data set.
- DSQ1BFRM creates QMF/GDDM charts and the QMF trace data set.

Load QMF/GDDM map sets to the GDDM ADMF data set

Important: This job replaces any existing QMF maps. Ensure that you back up ADMF if you want to keep any existing QMF maps.

1. Edit QMF810.SDSQSAPE(DSQ1EADM).
2. Verify that the installation parameters in the instream procedure of the job, and the job steps, match your tailoring specifications.

```
//DSQ1EADM PROC RGN='2048K',      Job-step region size
//         QMFTPRE='QMF810 ',   QMF prefix name for target libraries
//         GDDMADM='GDDM.ADMF' GDDM ADMF data set name
```

3. Submit QMF810.SDSQSAPE(DSQ1EADM).

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

Create QMF/GDDM charts and the QMF trace data set

If you are migrating to QMF Version 8.1 from a previous QMF release, skip this step.

DSQ1BFRM creates QMF/GDDM charts and the QMF trace data set.

1. Edit QMF810.SDSQSAPE(DSQ1BFRM).
2. Locate the installation parameters in the instream procedure of the job and ensure that they match your specifications.

```
//DSQ1BFRM PROC QMFTPRE='QMF810 ',     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 of DSQ1BFRM.
4. Tailor the VSAM control statement for your installations.

```
DEFINE CLUSTER (NAME(QMF810.DSQUCFRM) -
                  VOLUMES(QMFVOL) -     QMF/GDDM Charts volume
                  UNIQUE -
                  RECSZ(400 400) -
                  CONTROLINTERVALSIZE(2048) -
                  KEYS(20 0)) -
                DATA -
                  (RECORDS(1000 300)) -
                CATALOG(VSAMUSERCAT)    VSAM user catalog
```

5. Submit QMF810.SDSQSAPE(DSQ1BFRM).
6. Check for a return code of 0. If the return code is not 0, determine which steps ran correctly:
   - If some of DSQ1CFRM ran, edit DSQ1CFRM and remove the steps that ran successfully. Otherwise, you will receive error messages indicating that the objects are already there.
   - If all of DSQ1CFRM ran and the trace file is allocated, edit DSQ1BFRM and remove the last job step to create the QMF trace data set DSQDEBUG.

Translate, assemble and link-edit the QMF-supplied governor

Job DSQ1EGLK performs the translate, assemble, and link-edit for the QMF-supplied governor:

1. Edit job QMF810.SDSQSAPE(DSQ1EGLK) per the comments in the job.
2. Submit job QMF810.SDSQSAPE(DSQ1EGLK)
3. Check for a return code of 0 on all steps except LINKPROG, which can have a return code of 4. If the return code is not 0 or 4, correct the problem and rerun the job.

Update CICS control tables

Before you run QMF under CICS/ESA, you must describe QMF to CICS. To do this, you must modify both control table statements and a job that updates the CICS system definitions (CSDs).

CICS documentation is the authoritative source for information on how to set up the CICS tables. For details, see CICS/OS390 Resource Definition (Macro) and the CICS/OS390 Resource Definition (Online).

DCT (destination control table)

DSQ1CDCS and DSQ1CDCT in QMF810.SDSQSAPE describe the QMF trace data set to CICS.

1. Edit your CICS source for DFHDCT.
2. Find the local entry for TYPE=SDSCI and add a copy statement for DSQ1CDCS as shown in the following example:

```
*---------------------------------------------------------------
*    LOCAL ENTRIES FOR TYPE=SDSCI SHOULD BE PLACED BELOW THIS BOX
*---------------------------------------------------------------
          COPY DSQ1CDCS
```

3. Install the QMF trace facility.

   Find where local entries are specified and add a copy statement (DSQ1CDCT) for TYPE=EXTRA, as shown in the following example:

```
*---------------------------------------------------------------
*    OTHER LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
*---------------------------------------------------------------
          COPY DSQ1CDCT
```

4. Assemble and link-edit the member to create a new DFHDCT module.

Ensure the job completes with a return code of 0. If you receive higher return codes, check the list output and correct the error.

Update the CSD

DSQ1ECSD creates a new LIST called QMF, which is defined in the 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. DSQ1ECSD also defines the DB2ENTRY which is associated with the correct plan name and group by the QMF transaction. For other QMF - DB2 UDB for z/OS considerations, see “Describe QMF to DB2 in CICS” on page 35.

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

   The QMF panel data set requires a VSAM CI size of 32K. 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 32K, specify an LSRPOOL that supports a VSAM CI size of 32K through DFHCSDUP.

2. Edit QMF810.SDSQSAPE(DSQ1ECSD).
3. Verify or change the installation parameters in the instream procedure of the job to match your tailoring specifications.
4.
```
//DSQ1ECSD PROC REG=2048K,        Job Step Region
//      QMFTPRE='QMF810 ',         DSN Prefix for QMF
//      CLOAD='CICS.LOADLIB',     Name of CICS Program Lib
//      CCSD='CICS.DFHCSD',       Name of CICS CSD file
//      OUTC='*'                  Print sysout class
```
5. Submit the job and check that the job ran with a return code of 0. If you receive higher return codes, check the list output and correct the error.

Tailor the QMF profile

The ENVIRONMENT column of the Q.PROFILE table enables a single AUTHID to have different profiles depending on the environment (TSO or CICS). When installed under TSO, QMF initially assigns everything in the ENVIRONMENT column the value of NULL. Next, a new row is added with an AUTHID of SYSTEM and an ENVIRONMENT entry of CICS.

If you use the same AUTHID in CICS and TSO, and you use command synonyms that contain TSO commands, change all NULL entries to TSO entries as shown:

```
UPDATE Q.PROFILES SET ENVIRONMENT='TSO' WHERE ENVIRONMENT = NULL
```

After you issue this statement, QMF uses the SYSTEM row for the CICS environment.

Update CICS startup job stream

In this step, you update the DD statements that must be in the CICS startup job stream.

1. Ensure that the library containing the customized DSNHDECP for DB2 is accessible to z/OS through STEPLIB.

   ```
   //STEPLIB  DD DSN=CICS.SDFHAUTH,DISP=SHR
   //         DD DSN=DSN810.SDSNEXIT,DISP=SHR
   //         DD DSN=DSN810.SDSNLOAD,DISP=SHR
   ```

   You must individually APF-authorize each concatenated library.

   If your CICS release is 4.1 or later: DB2 does not need the DB2 program libraries in the DFHRPL DD statement. But, QMF does an EXEC CICS LOAD for DSNHDECP upon initialization, therefore, QMF requires that SDSNEXIT or SDSNLOAD (wherever your customized DSNHDECP module is located) be in the DFHRPL DD concatenation. Be sure to place these DB2 libraries after the CICS program libraries.

2. Place the load library containing QMF, GDDM, and DB2 UDB for z/OS modules in the CICS module load library list, DFHRPL.

```
//DFHRPL  DD  ...
//        DD  DSN=QMF810.SDSQLOAD,DISP=SHR
//        DD  DSN=GDDM.SADMMOD,DISP=SHR
//        DD  DSN=DSN.SDSNEXIT,DISP=SHR
//        DD  DSN=DSN.SDSNLOAD,DISP=SHR
```

   Be sure to use the correct DB2 UDB for z/OS release level when connecting from CICS (QMF loads DSNHDECP and DSNCLI).

3. Ensure access to the following data sets that 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=QMF810.DSQPNLE,DISP=SHR   QMF Panel File
//DSQDEBUG  DD DSN=QMF810.DSQDEBUG,DISP=SHR  Trace and Error Messages
//DSQUCFRM  DD DSN=QMF810.DSQUCFRM,DISP=SHR  User-Defined ICU Forms
```

4. Shut down and restart CICS to incorporate your changes to the CICS tables and to the CICS startup job. Continue on to Chapter 8, “Testing your QMF installation,” on page 51.

Chapter 6. Configuring remote servers for QMF Compatibility mode

This chapter explains how to install or migrate to QMF Compatibility mode in a remote server. A remote server is a DB2 server that is accessible from your local DB2 UDB for z/OS subsystem.

For each remote server that contains no previous release of QMF, proceed to one of the following sections:

- "Configuring QMF Compatibility mode as a DB2 for Linux, Unix® and Windows application"
- "Configuring QMF Compatibility mode as a DB2 UDB for iSeries Application" on page 44
- "Configuring QMF Compatibility mode as a DB2 UDB for z/OS application" on page 45

For each remote server that contains a previous release of QMF, proceed to "Migrating all server types to QMF Compatibility mode" on page 46.

Configuring QMF Compatibility mode as a DB2 for Linux, Unix® and Windows application

In this section all of the following DB2 products are referred to collectively as the DB2 DRDA AS: QMF support for DB2 distributed relational database architecture application servers is optional. You need to perform the steps described in this section only if you intend to connect QMF to any of the previously described DB2 DRDA application servers. When it is necessary, a more specific reference is made to one of the following products:

- DB2 Universal Database™ for Linux, Unix, and Windows Version 8.1
- DB2 DataJoiner® Version 2.1.1 or higher

QMF support for a DB2 DRDA AS is optional. You only need to perform the steps described in this section if you intend to connect QMF to any of the previously described DB2 DRDA Application Servers.

Before you install QMF into a DB2 DRDA AS from z/OS, you need to make the following preparations for DB2 DataJoiner:

- Create an installation ID on the DB2 DRDA AS and make it a member of the SYSADM GROUP.
- Create the database on the DB2 DRDA AS using the following command:

```
"db2 create database" database-name
```

Note: Normally you will want the created database to have authentication SERVER, which is the default. However, due to password handling restrictions with Microsoft SNA Server (on Windows NT), it is necessary to change the database to have authentication CLIENT. Refer to the appropriate *DB2 Command Reference* manual for the specific system commands to use for setting database authentication.

- On the DB2 DRDA AS, locally connect to the installation ID and verify that its authority level is SYSCRTL or SYSADM, using the commands:

```
"db2 connect to" database-name
                "user" system-id "using" password

     "db2 get authorizations"
```

- **Optional:** Grant additional administrative authorities to groups, users, or PUBLIC, as needed.

Check the step's completion code in the system messages. Completion messages can be found in the SYSTSPRT or the SYSTERM output, as indicated. SYSPRINT provides additional diagnostic information for IBM Software Support.

Installing QMF Compatibility mode

Follow the job sequence in the table below to install and prepare QMF Compatibility mode in a DB2 Universal Database for Linux, Unix and Windows Version 7.1 or later server that contains *no* previous release of QMF. To migrate from QMF Version 3.3 to QMF Compatibility mode, proceed to "Migrating all server types to QMF Compatibility mode" on page 46. If you want to install QMF New Function mode, see Chapter 7, "Configuring QMF New Function mode for all server types," on page 49.

Table 11. Job sequence for installing QMF Compatibility mode

Job name	Purpose
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1EDJ2	Creates QMF control tables
DSQ1BVW	Creates QMF Version 8.1 views
DSQ1BPKG	Binds QMF packages
DSQ1EDJ4	Optional: Creates QMF sample tables.

Starting QMF against a DB2 DRDA AS

Assuming you have started QMF under TSO or CICS, you should change the QMF parameters on your START command if you want to start QMF under DB2 DRDA AS. Specify the following command:

```
(DSQSSUBS=ssid,DSQSDBNM=location
```

where *ssid* is your DB2 UDB for z/OS subsystem ID and *location* is your DB2 DRDA AS location name.

You are ready to continue on to Chapter 8, "Testing your QMF installation," on page 51.

Deleting QMF from a DB2 DRDA AS

This section describes how to delete QMF from a DB2 DRDA AS.

Deleting QMF

Run this step only if you are reinstalling QMF into a DB2 DRDA AS application server that already contains QMF.

Attention: This step removes all QMF control tables and packages from the DB2 DRDA AS. QMF is not able to connect to the DB2 DRDA AS after running this step.

1. Edit QMF810.SDSQSAPE(DSQ1EDX1).
2. Read the job comments and change values as necessary.
3. Change *ssid* to your DB2 UDB for z/OS subsystem ID.
4. **Optional**: Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF810.SDSQSAPE(DSQ1EDX1).
6. Check Stepname DSQCDROP for a return code of 0 or 4. Review SYSTERM for completion messages.

   If the return code is other than 0 or 4, examine SYSTSPRT and SYSPRINT for error messages. Perform corrective actions and rerun the job.

Deleting QMF sample tables from a DB2 DRDA AS

Run this step only if you are reinstalling QMF into a DB2 DRDA AS application server that already contains QMF.

This step drops all the QMF sample tables from the DB2 DRDA AS.

1. Edit QMF810.SDSQSAPE(DSQ1EDX2).
2. Read the job comments and change values as necessary.
3. Change *ssid*to your DB2 UDB for z/OS subsystem ID.
4. (Optional) Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF810.SDSQSAPE(DSQ1EDX2).
6. Check Stepname DSQCDROP for a return code of 0 or 4. Review SYSTERM for completion messages.

   If the return code is other than 0 or 4, examine SYSTSPRT and SYSPRINT for error messages. Perform corrective actions and rerun the job.

Configuring QMF Compatibility mode as a DB2 UDB for iSeries Application

Follow the job sequence in the table below to install and prepare QMF Compatibility mode in a DB2 UDB for iSeries Version 5.1 or later server that contains *no* previous release of QMF. To migrate from QMF Version 3.3 to QMF Compatibility mode, proceed to "Migrating all server types to QMF Compatibility mode" on page 46. If you want to install QMF New Function mode, see Chapter 7, "Configuring QMF New Function mode for all server types," on page 49.

Table 12. Job sequence for brand new Compatibility mode installation into DB2 for iSeries

Job name	Purpose
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1EAS2	Creates QMF control tables
DSQ1BVW	Creates QMF Version 8.1 views
DSQ1BPKG	Binds QMF packages
DSQ1EAS4	Creates QMF sample tables

Starting QMF against a DB2 UDB for iSeries server

If you started QMF under TSO or CICS, change the QMF parameters on your START command to start QMF under DB2 Universal Database for iSeries.

Specify the following command:

```
(DSQSSUBS=ssid,DSQSDBNM=location
```

where *ssid* is your DB2 Universal Database for z/OS subsystem ID and *location* is your DB2 Universal Database for iSeries location name.

Deleting QMF

Run this step only if you are reinstalling QMF into a DB2 UDB for iSeries application server that already contains QMF.

Attention: This step removes all QMF control tables and packages from the DB2 UDB for iSeries AS. QMF is not able to connect to the DB2 UDB for iSeries AS after running this step.

1. Edit QMF810.SDSQSAPE(DSQ1EDX1).
2. Read the job comments and change values as necessary.
3. Change *ssid* to your DB2 UDB for z/OS subsystem ID.
4. (Optional) Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF810.SDSQSAPE(DSQ1EDX1).

6. Check Stepname DSQCDROP for a return code of 0 or 4. Review SYSTERM for completion messages.

   If the return code is other than 0 or 4, examine SYSTSPRT and SYSPRINT for error messages. Perform corrective actions and rerun the job.

Deleting QMF sample tables from a DB2 UDB for iSeries AS

This step should be run only if you are reinstalling QMF into a DB2 UDB for iSeries AS that already contains QMF.

This step drops all the QMF sample tables from the DB2 UDB for iSeries AS.

1. Edit QMF810.SDSQSAPE(DSQ1EDX2).
2. Read the job comments and change values as necessary.
3. Change <ssid> to your DB2 UDB for z/OS subsystem ID.
4. (Optional) Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF810.SDSQSAPE(DSQ1EDX2).
6. Check Stepname DSQCDROP for a return code of 0 or 4. Review SYSTERM for completion messages.

   If the return code is other than 0 or 4, examine SYSTSPRT and SYSPRINT for error messages. Perform corrective actions and rerun the job.

Configuring QMF Compatibility mode as a DB2 UDB for z/OS application

This chapter addresses installing QMF into a DB2 UDB for OS/390 Version 6.1 or later server that is accessed through a local DB2 UDB for z/OS requester.

Follow the job sequence in the table below to install and prepare QMF Compatibility mode in a DB2 UDB for OS/390 Version 6.1 or later server that contains *no* previous release of QMF. To migrate from QMF Version 3.3 to QMF Version 8.1 Compatibility mode, proceed to “Migrating all server types to QMF Compatibility mode” on page 46. If you wan to install QMF Version 8.1 New Function mode, see Chapter 7, “Configuring QMF New Function mode for all server types,” on page 49.

Table 13. Job sequence to install Compatibility mode in a DB2 UDB for z/OS server with no previous release of QMF

Job name	Purpose
DSQ1TBAJ	Optional: Creates QMF VCAT name
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1TBLJ	Creates QMF control tables
DSQ1BVW	Creates QMF Version 8.1 views
DSQ1STGJ	Creates table space for QMF IVP

Table 13. Job sequence to install Compatibility mode in a DB2 UDB for z/OS server with no previous release of QMF (continued)

Job name	Purpose
DSQ1EIVS	Optional: Creates QMF sample tables

Migrating all server types to QMF Compatibility mode

This section will address migrating your server from QMF versions 3.3, 6.1, 7.1, or 7.2 to a QMF Compatibility mode installation. Servers that are supported for this installation are:

- DB2 UDB for OS/390 Version 6.1 and later
- DB2 UDB for Linux, Unix, and Windows Version 8.1 and later
- DB2 UDB for Unix and Windows Version 7.1
- DB2 UDB for iSeries Version 5.1 and later
- DB2 Server for VM/VSE Version 7.1 and later
- DB2 DataJoiner Version 2.1.1 and later

If QMF Version 3.3 or later exists at the server, run the following job steps to perform the migration. Note that the previous level of QMF will still be functional. Running the migration steps below will allow the previous level of QMF and DB2 QMF Version 8.1 to coexist in the same server. QMF sample tables from previous releases of QMF are still valid with DB2 QMF Version 8.1.

Table 14. Job sequence for migration from QMF versions 3.3, 6.1, 7.1, and 7.2 to QMF Compatibility mode installation into a remote database server (DB2 UDB for z/OS, DB2 UDB for Linux, Unix, and Windows, DB2 UDB for Unix, Windows, and OS2, DB2 UDB for iSeries, and DB2 Server for VM/VSE)

Job name	Purpose
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1BVW	Creates QMF Version 8.1 views
DSQ1BPKG	Binds QMF packages

You also use the previous job sequence path in the table above to migrate server levels. For example, if you install DB2 QMF Version 8.1 into DB2 UDB for OS/390 Version 6 and DB2 to DB2 UDB for z/OS and OS/390 Version 7. The steps in the table above address what needs to be done for DB2 QMF Version 8.1 after the server migration. These same steps must be run if a DB2 UDB for z/OS Version 8 database is migrated from Compatibility to New Function mode.

Previous releases of QMF sample tables are still valid in DB2 QMF Version 8.1. If the older sample tables were not previously installed and you want to install them now with DB2 QMF Version 8.1, run the appropriate job in the table below. Jobs are also included to delete the old sample tables. You can reinstall the deleted sample tables if you want.

Table 15. Jobs that delete or install sample tables

Job Name	Purpose
DSQ1EDSJ	Deletes sample tables for DB2 UDB for z/OS servers
DSQ1EDX2	Deletes sample tables for DB2 UDB for Unix, Windows, and OS/2; DB2 DataJoiner, and DB2 UDB for iSeries servers
DSQ1EVS	Installs sample tables for DB2 UDB for z/OS servers
DSQ1EDJ4	Installs sample tables for DB2 UDB for Unix and Windows, and DB2 DataJoiner servers
DSQ1EAS4	Installs sample tables for DB2 UDB for iSeries servers

Configuring remote servers for Compatibility mode

Chapter 7. Configuring QMF New Function mode for all server types

These instructions will help you configure QMF New Function mode in a supported server that contains *no* previous release of QMF, or in a supported server that contains QMF Compatibility mode.

A QMF New Function mode supported server type can be one of the following:

- DB2 UDB for z/OS Version 8.1 in NFM
- DB2 UDB for Linux, Unix, and Windows Version 8.1 or later
- DB2 UDB for iSeries Version 5.1 and later
- DB2 UDB for Unix and Windows Version 7.1

Installing QMF New Function mode

A QMF New Function mode installation must be the only version of QMF on the server.

Table 16. Job sequence for new NFM install into any server

Job name	Purpose
DSQ1TBAJ	Optional: Creates QMF VCAT (DB2 UDB for z/OS servers only)
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1BLNI	Creates QMF NFM control tables
DSQ1BVW	Creates DB2 QMF V8.1 views
DSQ1BPKG	Binds QMF packages
DSQ1BINR	DB2 UDB for z/OS requesters only: binds the QMF application plan
DSQ1STGJ	DB2 UDB for z/OS servers only: creates the QMF SAVE DATA table space for QMF IVP
DSQ1EIVS	DB2 UDB for z/OS servers only: creates QMF sample tables
DSQ1EDJ4	DB2 UDB for Unix and Windows servers only: creates QMF sample tables
DSQ1EAS4	DB2 UDB for iSeries only: creates QMF sample tables

Migrating to QMF New Function mode

This series of steps will migrate a QMF Compatibility mode installation to a QMF New Function mode installation. **Previous releases of QMF cannot be used in this server after this migration is performed.**Once the server has migrated from QMF Version 8.1 Compatibility mode to QMF Version 8.1 New Function mode, there is no fallback.

Table 17. Job sequence to migrate from QMF Compatibility mode to QMF New Function mode

Job name	Purpose
DSQ1BSQL	Binds QMF installation packages and plan
DSQ1BLNM	Migrates QMF control tables to QMF NFM
DSQ1BROG	DB2 UDB for z/OS servers only: sample job to REORG the QMF table spaces. You can use your own utilities to perform this function.
DSQ1BINX	Drops and creates QMF indexes
DSQ1BVW	Creates DB2 QMF V8.1 views
DSQ1BPKG	Binds QMF packages

Previous releases of QMF sample tables are still valid in DB2 QMF Version 8.1. If the older sample tables were not previously installed and you want to install them with DB2 QMF Version 8.1, run the appropriate job in the table below. Jobs are also included to delete the old sample tables. You can reinstall the deleted sample tables if you wish.

Table 18. Jobs that delete or install sample tables

Job name	Purpose
DSQ1EDSJ	Deletes sample tables for DB2 UDB for z/OS servers
DSQ1EDX2	Deletes sample tables for DB2 UDB for Unix, Windows and OS/2; DB2 DataJoiner, and DB2 UDB for iSeries servers
DSQ1EVS	Installs sample tables for DB2 UDB for z/OS servers
DSQ1EDJ4	Installs sample tables for DB2 UDB for Unix, Windows and OS/2; and DB2 DataJoiner servers
DSQ1EAS4	Installs sample tables for DB2 UDB for iSeries servers

Chapter 8. Testing your QMF installation

This chapter describes the final steps in the install process.

The chapter includes the steps:
- Run the IVP (TSO)
- Run the IVP (CICS)
- Install the QMF application queries and application objects (TSO)
- Run the batch-mode IVP (optional)
- Clean up after installation

Run the IVP (TSO)

This step leads you through the final testing of QMF, called the installation verification procedure (IVP). To test a DB2 QMF for TSO/CICS installation, you need to start QMF, connect to your server or DB2 UDB for z/OS database, validate the existence of Help panels, and issue the command:

```
LIST TABLES (OWNER=Q)
```

. Most of the QMF product installation is tested by simply starting QMF. If you plan to run QMF in batch mode, there is a separate IVP, which follows the interactive IVP.

1. Complete all the installation and tailoring for the base product, as outlined in this book.
2. Ensure you have proper authority.

   If you start the QMF transaction with the authorization ID of Q, you already have the necessary DB2 UDB for z/OS authority. If you do not use the authorization ID Q, you need, at a minimum, the authority granted by the following SQL statements:

   ```
   GRANT SELECT ON Q.PROFILES TO authid
   GRANT SELECT ON Q.ERROR_LOG TO authid
   GRANT ALL ON Q.OBJECT_DIRECTORY TO authid
   GRANT ALL ON Q.OBJECT_DATA TO authid
   GRANT ALL ON Q.OBJECT_REMARKS TO authid
   ```

   where *authid* is your primary authorization ID.

   You must also have enough DB2 UDB for z/OS authority to exercise the IVP's SAVE DATA command. If you created the receiving database and table space, you already have this authority. If not, you need, at a minimum, the authority granted by the following SQL statements:

```
GRANT CREATETAB ON DATABASE dbname TO authid
GRANT USE OF TABLE SPACE dbname.table space TO authid
```

where *dbname* is the database name, *table space* is the table space name, and *authid* is your primary authorization ID.

If you chose the default values when you created the table space and database, the database is named DSQDBDEF, and the table space DSQTSDEF. If not, the names might be from the IVP on an earlier QMF release.

3. Start QMF

   Use the logon procedure or CLIST to invoke QMF, as in "Start QMF" on page 30.

   The QMF Home panel displays.

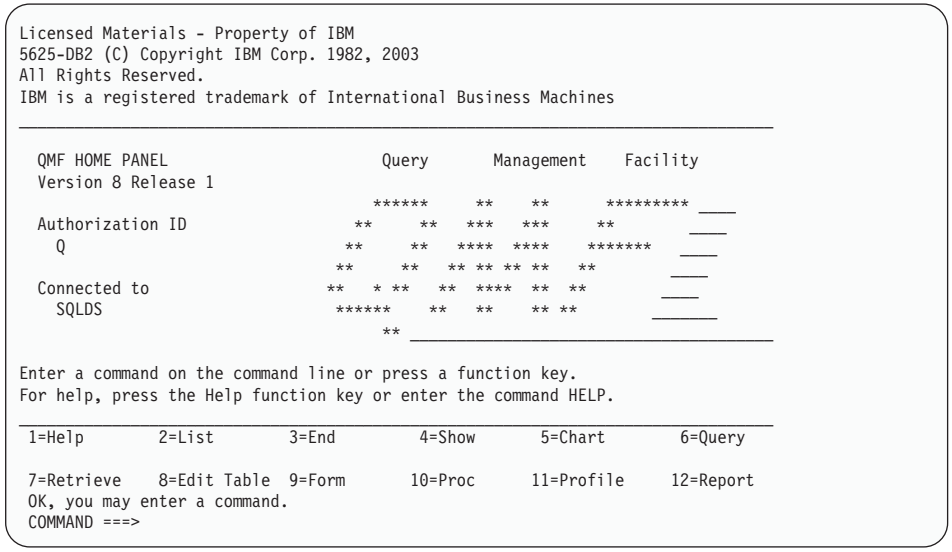


Figure 6. QMF Home panel

If the location name has not been defined for the database, *Connected to <location_name>* will not appear on the QMF Home panel.

Be sure you are connected to the Workstation Database Server or DB2 UDB for z/OS database in which you just installed QMF. If necessary, you can use the QMF CONNECT command to connect to the correct location.

If QMF does not start correctly, you might receive an error message. See Appendix A, "Miscellaneous," on page 357 for descriptions of common error conditions and corrective actions. Correct the problem and begin the IVP again.

4. Press the Help function key from the Home panel to validate the existence of help panels.
5. Exit from the help panel by pressing either F3 or F12.
6. Obtain a list of QMF-supplied sample tables.

   Type the QMF command `LIST TABLES (OWNER=Q)` on the command line and press Enter.

   If you press F8, additional panels are shown. Return to the QMF Home panel by pressing the Cancel function key. End the QMF session by pressing F12.
7. Run the QMF Installation Verification Procedure

   Replace QMF810 with the actual prefix you used for the QMF data sets, then issue this command from the QMF command line:

   ```
   IMPORT PROC FROM'QMF810.SDSQSAPE(DSQ1EIVP)'
   ```

   This will display the Installation Verification Procedure; comments at the beginning tell you how to run it. Before you run the IVP, replace every occurrence of QMF810 with the actual prefix that you used for the QMF data sets. If the procedure does not run successfully, use the QMF messages and message help panels to fix the problem.

The installation verification for interactive mode is now complete.

Run the IVP (CICS)

This step leads you through the final testing of QMF, called the installation verification procedure (IVP). To test that you have installed DB2 QMF for TSO/CICS properly, you need to start QMF, connect to your database, and verify that the QMF trace facility is installed. Most elements of the QMF product installation are tested by starting QMF.

Before you start QMF

1. Complete all the installation and customization steps outlined in this book.
2. Start the database connection, if it has not already started.
3. Verify that the QMF Trace Facility is installed by checking the transient data queue (DSQD). From a clear CICS screen, enter:

   ```
   CEMT INQUIRE QUEUE(DSQD)
   ```

   You should see a screen similar to this:

```
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, you need to review your modifications to the CICS DCT. Verify that the QMF trace file was installed correctly. See "Update CICS control tables" on page 37 for details.

Start and test QMF

This procedure starts the DB2 QMF for TSO/CICS product and tests that the product is properly installed. If you receive an error message during any part of the procedure, it indicates that QMF did not start properly. Under these circumstances, start by investigating some of the more common problems as described in Appendix A, "Miscellaneous," on page 357.

1. Sign on to the CICS system that is connected to QMF.
2. Press the Escape function key to begin a native CICS session.
3. Start QMF by issuing the CICS transaction, `QMFE`. Also specify the use of the temporary storage queue (DSQSDBQT) so that you can view any warning messages online. To start QMF with the temporary storage queue name, DSQD, specify:

   ```
   QMFE DSQSDBQT=TS,DSQSDBQN=DSQD
   ```

   You should see the QMF Home panel.

```
Licensed Materials - Property of IBM
5675-DB2 5697-F42 (C) Copyright IBM Corp. 1982, 2002
All Rights Reserved.
IBM is a registered trademark of International Business Machines
_____________________________________________________________________________

  QMF HOME PANEL                        Query       Management    Facility
  Version 8 Release 1
                                      ******     **    **      ********* ____
  Authorization ID                   **     **   ***   ***     **         ____
    Q                                **     **   ****  ****    *******    ____
                                    **     **   ** ** ** **   **          ____
  Connected to                     **   * **   **  ****  **  **           ____
    SQLDS                           ******    **   **    ** **          ______
                                         **  _____________________________________

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 ===>
```

4. Verify the existence of QMF online help.
   Press the Help function key. You should see this Help panel:

```
_____________________________________________________________________________
Licensed Materials - Property of IBM
5645-DB2 5648-A70 (C) Copyright IBM Corp. 1982, 2002
All Rights Reserved.
IBM is a registered trademark of International Business Machines
+-----------------------------------------------------------------------------+
|                        Help: Query Management Facility                      |
|                                                                             |
| Select a topic.                                                             |
|                                                                1  to 7  of 14 |
|     1. What's new in Version 8                                              |
|     2. Profile                                                              |
|     3. QMF commands                                                         |
|     4. Prompted Query                                                       |
|     5. SQL (Structured Query Language)                                      |
|     6. Table Editor                                                         |
|     7. Forms                                                                |
+-----------------------------------------------------------------------------+
| F1=Help  F3=Exit  F7=Backward  F8=Forward  F9=Keys  F12=Cancel              |
+-----------------------------------------------------------------------------+


OK, HELP performed. Please proceed.
```

   Exit from the Help panel by pressing either PF3 or PF12.
5. Obtain a list of QMF-supplied sample tables.

Type the QMF command `LIST TABLES (OWNER=Q)` on the command line and press Enter. Depending on whether you previously installed QMF, the tables that have the owner Q might vary from the following screen:

```
+------------------------------------------------------------------------+
|                                Table List                              |
|                                                                        |
| Action   Name               Owner                                      |
|                                                      1  to  7 of 36    |
|          APPLICANT          Q                                          |
|          COMMAND_SYNONYMS   Q                                          |
|          DSQ_RESERVED       Q                                          |
|          DSQEC_ALIASES      Q                                          |
|          DSQEC_COLS_LDB2    Q                                          |
|          DSQEC_COLS_RDB2    Q                                          |
|          DSQEC_QMFOBJS      Q                                          |
|          DSQEC_TABS_LDB2    Q                                          |
|          DSQEC_TABS_RDB2    Q                                          |
|          INTERVIEW          Q                                          |
|          ORG                Q                                          |
|          PARTS              Q                                          |
+------------------------------------------------------------------------+
| F1=Help  F4=Command  F5=Describe  F6=Refresh  F7=Backward  F8=Forward  |
| F9=Clear  F10=Comments  F11=Sort  F12=Cancel                           |
+------------------------------------------------------------------------+
OK, your database object list is displayed.
```

If you press PF8, additional panels are shown. Return to the QMF Home panel by pressing the Cancel function key. End the QMF session by pressing PF12.

The installation verification is now complete. You can browse the temporary storage queue to determine if there are any QMF warning messages using the CICS transaction:

```
CEBR DSQD
```

If your IVP ran without any errors, your TS queue DSQD should be empty.

Install the QMF application queries and procedures (TSO)

This step updates sample queries and procedures for QMF applications. These applications include the Displaying Printed Reports (DPRE), layout, and the document interface. The optional batch IVP uses these sample queries and procedures as part of the test.

After QMF is successfully installed and tested, you can use it to create the QMF-supplied sample queries, procedures, and command synonyms.

Complete this step by running one or two QMF procedures:

Procedure
Description

DSQ1ESQD
: Deletes the sample queries and procedures from a previous QMF release

DSQ1ESQI
: Adds the new sample queries and procedures to the QMF database

1. Delete the current sample queries and procedures.

   If there is no existing QMF release on the system, or if the previous release is in another DB2 UDB for z/OS subsystem, skip this step.

   a. Begin a QMF session.
   b. Connect to the Workstation Database Server or DB2 UDB for z/OS server in which you just installed QMF.
   c. Enter the following command from within QMF:

      ```
      IMPORT PROC FROM 'QMF810.SDSQSAPE(DSQ1ESQD)'
      ```

      where *QMF810* is the prefix for the QMF data sets. If you used another prefix, change the name accordingly.
   d. Run the procedure.
2. Add the sample queries and procedures to your QMF database.

   Enter the following command in a QMF session:

   ```
   IMPORT PROC FROM 'QMF810.SDSQSAPE(DSQ1ESQI)'
   ```

   where *QMF810* is the prefix for the QMF data sets. If you used another prefix, change the name accordingly.
3. Check that you receive a message that says the objects were installed correctly.

   If a failure occurs, rerun the first execution step to delete any partially created objects. Then run the second step.

Optional: Run the batch-mode IVP

Skip this step if your installation does not use batch-mode QMF.

This step tests batch-mode IVP by running the batch-mode job that you created in "Set up a QMF batch job to run batch IVP (optional)" on page 34. The job begins a background TSO session in which QMF runs the procedure Q.DSQ1EBAT. The procedure conducts the batch-mode IVP and tests the following batch-mode operations:

- Reaching and starting QMF
- Importing, saving, running, and erasing a query
- Saving, retrieving, and erasing a new table
- Printing a query
- Exporting a query, then erasing it with the QMF TSO command

The IVP is successful when it runs without error and prints the following query:

```
DELETE FROM &NAME
   WHERE OWNER = USER AND NAME = 'QMF_IVPQUERY'
```

1. Examine the JCL.

   The resources QMF needs in batch mode and those it needs interactively are basically the same. You can create the batch job out of the sample TSO logon procedure. Be certain that your batch job allocates DSQPRINT. Output from the QMF PRINT command goes into this file.

2. Examine the QMF procedure Q.DSQ1EBAT.

   You created Q.DSQ1EBAT with the sample queries and procedures. It was saved with SHARE=YES. You can therefore examine and edit it on the screen. If you are not using QMF810 as the prefix for the QMF data sets, you must change the procedure's IMPORT commands, which retrieve queries from the QMF sample library.

   If you change the procedure, you must save it under your own logon ID; be sure to specify SHARE=YES. If you started QMF as an ISPF dialog, you must change the ISPSTART statement in the batch IVP JCL to reflect the new ownership of the procedure. For example, if your logon ID is JONES, the modified statement looks like this:

   ```
   ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSMODE=B,DSQSRUN=JONES.DSQ1EBAT)
   ```

3. Run the job.

4. Check the printed output, the table Q.ERROR_LOG, and the DSQDEBUG data set for errors. If an error is recorded in Q.ERROR_LOG or DSQDEBUG, you can use the HELP command to see the corresponding message help panel.

   If the job fails, you can correct the error and rerun it.

Clean up after installation

If you do not have a previous release of QMF installed, skip this step.

Attention: This step removes your previous release of QMF. Do not perform this step until you are certain that the earlier version is no longer needed.

Choose one of these procedures:

- Freeing an earlier application plan

  This step removes the previous release when QMF Version 8.1 and the release are in the same DB2 UDB for z/OS subsystem.
- QMF Version 8.1 and a previous release are in different DB2 UDB for z/OS subsystems

  This step removes the previous release when QMF Version 8.1 and the release are in different DB2 UDB for z/OS subsystems.

After running either of these two substeps, you can delete the libraries of the previous QMF release. Table 19 lists these libraries with their default prefixes. The names at your installation might not be the ones shown.

Attention: Pay special attention to the prefix to avoid deleting a Version 8.1 data set.

Table 19. Libraries to be deleted from earlier QMF releases

V3RxMy data sets	V6R1 data sets	V7R1M0 data sets	V7R2M0 data sets
QMF3xy.ADMFE	QMF610.SDSQCLTE	QMF710.SDSQCLTE	QMF720.SDSQCLTE
QMF3xy.CICS.DFHTEMP	QMF610.SDSQEXCE	QMF710.SDSQEXCE	QM720.SDSQEXCE
QMF3xy.DSQPMSE	QMF610.SDSQMLBE	QMF710.SDSQMLBE	QMF720.SDSQMLBE
QMF3xy.DSQDBRMD	QMF610.SDSQPLBE	QMF710.SDSQPLBE	QMF720.SDSQPLBE
QMF3xy.DSQSAMPE	QMF610.SDSQSAPE	QMF710.SDSQSAPE	QMF720.SDSQSAPE
QMF3xy.SDSQMAPE	QMF610.SDSQSLBE	QMF710.SDSQSLBE	QMF720.SDSQSLBE
QMF3xy.DSQCLSTE	QMF610.SDSQUSRE	QMF710.SDSQUSRE	QMF720.SDSQUSRE
QMF3xy.DSQEXECE	QMF610.SDSQLOAD	QMF710.SDSQLOAD	QMF720.SDSQLOAD
QMF3xy.DSQUSERE	QMF610.SDSQDBRM	QMF710.SDSQDBRM	QMF720.SDSQDBRM
QMF3xy.DSQPLIBE	QMF610.SDSQMAPE	QMF710.SDSQMAPE	QMF720.SDSQMAPE
QMF3xy.DSQSLIBE	QMF610.SDSQCHRT	QMF710.SDSQCHRT	QMF720.SDSQCHRT
QMF3xy.DSQMLIBE	QMF610.DSQPVARE	QMF710.DSQPVARE	QMF720.DSQPVARE
QMF3xy.DSQLOAD	QMF610.DSQPNLE	QMF710.DSQPNLE	QMF720.DSQPNLE
QMF3xy.DSQDBRM	QMF610.ADSQOBJ	QMF710.ADSQOBJ	QMF720.ADSQOBJ
QMF3xy.DSQTLIBE	QMF610.ADSQDBMD	QMF710.ADSQDBMD	QMF720.ADSQDBMD
QMF3xy.SDSQCHRT	QMF610.ADSQMACE	QMF710.ADSQMACE	QMF720.ADSQMACE
QMF3xy.DSQMACE	QMF610.ADSQPMSE	QMF710.ADSQPMSE	QMF720.ADSQPMSE
QMF3xy.DSQOBJ	QMF610.DSQDEBUG	QMF710.DSQDEBUG	QMF720.DSQDEBUG
QMF3xy.DSQPNLE			QMF720.SDSQEXIT
QMF3xy.DSQPVARE			
QMF3xy.DSQUCFRM			

Freeing an earlier application plan

Run this step only when QMF Version 8.1 and the previous release are on the same DB2 UDB for z/OS subsystem.

1. Edit QMF810.SDSQSAPE(DSQ1JFPL).

2. Change the job statement to conform to your site's conventions.
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='QMF810',             QMF prefix
//     DB2EXIT='DSN810.SDSNEXIT',  DB2 Universal Database for
//                                       z/OS exit library
//     DB2LOAD='DSN810.SDSNLOAD'   DB2 Universal Database for
//                                       z/OS program library
```

4. Edit QMF810.SDSQSAPE(DSQ1DEL1).
5. Replace DSN with the name of the DB2 Universal Database for z/OS subsystem, and replace QMF810 with the name of the application plan of the previous release.

```
DSN SYSTEM(DSN)
FREE PLAN(QMF810)
```

Table 20. QMF release defaults

Previous Release	Default
QMF Version 7.2	QMF720
QMF Version 7.1	QMF710
QMF Version 6.1	QMF610
QMF Version 3.3	QMF330

6. Submit the job QMF810.SDSQSAPE(DSQ1JFPL).

   If the job fails, correct the error and rerun the job.

Deleting QMF Version 7.2 and earlier from a DB2 subsystem

Run this step only if DB2 QMF Version 8.1 and the earlier release are on different DB2 UDB for z/OS subsystems. The step frees the earlier application plan and drops various DB2 UDB for z/OS entities that belong to the earlier QMF release.

Attention: This job removes all traces of QMF from the DB2 UDB for z/OS subsystem and should be run only if the current release of QMF does not exist in the DB2 UDB for z/OS subsystem.

1. Edit QMF810.SDSQSAPE(DSQ1DELA).
2. Change the job statement to conform to your site's conventions.
3. 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='QMF810',             QMF prefix
// DB2EXIT='DSN810.SDSNEXIT',    DB2 Universal Database for z/OS
//                                   exit library
// DB2LOAD='DSN810.SDSNLOAD'     DB2 Universal Database for z/OS
//                                program library
```

4. Edit member QMF810.SDSQSAPE(DSQ1DEL1).
5. Replace DSN with the name of the DB2 UDB for z/OS subsystem and replace QMF810 with the name of the application plan of the previous release.

```
DSN SYSTEM(DSN)
FREE PLAN(QMF810)
```

Table 21. QMF release defaults

Previous Release	Default
QMF Version 7.2	QMF720
QMF Version 7.1	QMF710
QMF Version 6.1	QMF610
QMF Version 3.3	QMF330

6. Edit member QMF810.SDSQSAPE(DSQ1DEL2).

   This member contains SQL statements to drop views, table spaces, databases, and storage groups.

   If the previous QMF release does not have the receiving table space for the users' SAVE DATA commands and the IVP, delete the following statement:

```
DROP STOGROUP DSQSGDEF
```

   QMF810

7. Edit member QMF810.SDSQSAPE(DSQ1DEL13)

   This member contains statements to delete user managed data sets for QMF control tables. There is no need to run this step if it is managed by DB2.

8. Submit job QMF810.SDSQSAPE(DSQ1DELA).

   If the job fails, correct the error and rerun the job.

Part 2. Managing QMF for TSO/CICS

Chapter 9. Starting QMF

This chapter describes the different ways you can start QMF.

For information about starting QMF from the callable interface, see *Developing DB2 QMF Applications*.

Setting up and starting QMF on z/OS

On z/OS, QMF can be set up to run under TSO, ISPF, as a batch job, or under CICS.

Choosing an authorization ID on z/OS

A QMF session is started and runs under the authorization ID of the user or program that starts the QMF session.

You can assign a single SQL authorization ID or a single primary authorization ID with one or more secondary authorization IDs.

The SQL authorization ID must be either a primary or secondary authorization ID. Both the primary and secondary IDs are fixed for the duration of the user's session.

Authorization IDs can be as long as your DB2 database will allow. The first character must be a letter, and the remaining seven characters can consist of letters or numbers. For rules of these names, see the *DB2 UDB for z/OS SQL Reference*. Authorization IDs are the source of all DB2 privileges. Each authorization ID can possess any number and any kind of DB2 privileges. For example, JONES is one of the user A's authorization IDs, and JONES has the SELECT privilege on the table SMITH.TABLEA. Thus, user A also has the SELECT privilege on SMITH.TABLEA, and can run a SELECT query on that table.

Setting up QMF to run in native z/OS as a batch job

Allow your users to start QMF in native z/OS as a batch job by creating JCL that defines where the spill file is, where the panels are stored, the file names of the panels, and the names and locations of other tables and QMF objects.

To issue a QMF command, specify the name of an initial QMF procedure. In Figure 7 on page 68, the name of the procedure is I=X, where X is the name of the QMF procedure.

QMF starts and runs procedure X. When procedure X completes, QMF terminates. The QMF return code is returned in register 15. You can test it using the standard JCL condition code testing.

```
//RUNQMF  EXEC PGM=DSQQMFE,PARM='M=B,I=X,P=QMF810,S=DSN'
//*********************************************************
//*     Program load libraries
//*********************************************************
//STEPLIB  DD DSN=QMF810.SDSQLOAD,DISP=SHR
//         DD DSN=DSN810.SDSNEXIT,DISP=SHR
//         DD DSN=DSN810.SDSNLOAD,DISP=SHR
//         DD DSN=GDDM.GDDMLOAD,DISP=SHR
//*********************************************************
//*     QMF/GDDM maps
//*********************************************************
//ADMGGMAP DD DSN=QMF810.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)
//DSQUDUMP DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//DSQSPILL DD DSN=&&SPILL,DISP=(NEW,DELETE),
// UNIT=SYSDA,SPACE=(TRK,(100),RLSE),
// DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
```

Figure 7. 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.

Setting up and starting QMF on TSO

DD statements in a logon procedure can allocate resources for the user.

Using DD statements in the logon procedure

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 user and terminal during a TSO session. If your installation uses its own TMP, rather than one supplied by IBM, some of the following information 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. In particular, you can allocate a data set that is unique to the user.

The following statement within a CLIST allocates a unique library for its user's CHART forms. The name of the allocated library begins with the user's TSO logon ID, represented by the variable &SYSUID:

```
ALLOC DDNAME(DSQUCFRM) DSNAME('&SYSUID...CHARTLB.DATA') OLD
```

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 need to ensure that the program load libraries, modules, and data sets are available to QMF, and that GDDM and DB2 requirements are met.

Defining a TSO ID

When you start QMF under TSO, you assign authorization IDs through the DB2 exit routine, DSN3@ATH (IBM also supplies a default exit routine). If the user's TSO logon has been passed to it, the routine returns a list of the assigned authorization IDs. If you want to use the default exit routine DSN3@ATH without changes:

- A users's primary and SQL authorization IDs match the user's TSO logon ID
- No secondary authorization IDs are assigned

TSO considerations

Use whichever ddname is established by your installation 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. Figure 8 lists the data sets used by TSO. If you do not know your installation'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=        QMF810.SDSQCLTE,DISP=SHR
//SYSEXEC DD DSN=SYS2.EXEC,DISP=SHR
//        DD DSN=QMF810.SDSQEXCE,DISP=SHR
//SYSHELP DD DSN=SYS1.HELP,DISP=SHR
//EDT     DD DSN=&EDIT,UNIT=SYSDA,SPACE=(1668,(40,12))
```

Figure 8. Data sets used by TSO

Starting QMF with the TSO CALL command

You can also use the TSO CALL command to start QMF. Specify the name of the QMF load library, and pass the optional program parameters following the data set name, as in the following example:

```
CALL 'QMF810.SDSQLOAD(DSQQMFE)' 'DSQSMODE=I,DSQSSUBS=DB2T'
```

The QMF load library becomes a TASKLIB for the duration of the CALL command. However, you need to give QMF access to the DB2 and GDDM libraries in order to LOAD program interfaces to those products. In most

cases, DB2 and GDDM libraries are not part of TASKLIB. IF DB2 and GDDM libraries are not available, QMF terminates with an error.

Starting QMF directly with the DSQQMFE module

You can start QMF under TSO by entering DSQQMFE either from the command line in the READY mode, or in a CLIST or exec:

```
DSQQMFE DSQSBSTG=123456,DSQSDBUG=ALL,DSQSIROW=0,DSQSRUN=SAM.PROG1
```

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 Server error occurred

Starting QMF in a batch environment

To start QMF without using ISPF services, place the following statement in the SYSTSIN data set of your JCL on z/OS:

```
DSQQMFE  ...DSQSMODE=B,DSQSRUN=aaa.bbb
```

where DSQSMOD=B establishes the appropriate operating mode, and DSQSRUN=aaa.bbb identifies the procedure to be run. The procedure can include a variable as the procedure name; it should contain the authorization ID of the owner.

The ellipsis represents optional parameter values that the user can include in addition to the required DSQSMODE and DSQSRUN parameters.

Examples of starting QMF under TSO

The following examples show how to start and pass parameters to QMF operating independently of ISPF. The first two statements for TSO turn on L2 tracing (DSQSDBUG=NONE), pass a value of 50,000 for DSQSBSTG (maximum storage for reports), and pass a value of B (batch) for DSQSMODE (mode of operation).:

- Starting from TSO READY mode:

  ```
  DSQQMFE DSQSBSTG=50000,DSQSDBUG=NONE,DSQSMODE=B
  ```

- Starting from a CLIST or exec and specifying an initial procedure:

  ```
  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
  – To 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
```

Setting up and 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:

- 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 (z/OS).

You can use any of the previous methods to start 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 or a TSO command is used, the results can be unpredictable.

Restrictions:

1. You cannot run QMF as a command dialog. For example, the following statements are not valid:

   ```
   ISPEXEC SELECT CMD(DSQQMFE) NEWAPPL(DSQE)
   ISPSTART CMD(DSQQMFE) NEWAPPL(DSQE)
   ```

2. 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

If you choose to set up a menu option to start QMF, the menu must point to QMF. Figure 9 on page 72 shows a sample definition for the ISPF master application menu and 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 ABC 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(ISP0PRIM) NEWAPPL'
                2,'PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB PARM(DSQSSUBS=ABC)'
            /*                              */
            /* ADD Other APPLICATIONS HERE */
            /*                              */
                P,'PANEL(ISPOPT)'
                X,'EXIT'
                ' ',' '
                  *,'?' )
)END
```

Figure 9. Sample Master Application menu

The direct menu approach to starting QMF can be faster than the CLIST approach. If you allocate all user resources through TSO logon procedures, then the CLIST that you create for the menu option has no resources to allocate. The CLIST is left with a single function, starting QMF, which can be run without a CLIST.

You can add more than one option to your menu. Suppose, for example, that ABC is an experimental DB2 subsystem and DSN is the production subsystem. You can now 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 added lines in the menu's PROC section might look like this:

```
2,'PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB PARM(DSQSSUBS=DB2SSFDX)'
3,'PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB PARM(DSQSSUBS=DB2SSFDY)'
```

Using LIBDEF statements on z/OS

You may optionally use the ISPF LIBDEF statements to allocate QMF libraries during an ISPF session.

Allocate the program libraries to a unique QMF DD NAME of DSQLLIB to use the ISPF LIBDEF service for QMF and DB2 programs. Then specify DD NAME DSQLLIB as the ID value in the LIBRARY option on 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('QMF810.SDSQEXIT,'QMF810.SDSQLOAD',
'DSN810.SDSNEXIT','DSN810.SDSNLOAD') SHR REUSE
LIBDEF ISPLLIB LIBRARY ID(DSQLLIB)
```

To allocate program libraries using the ISPF LIBDEF service, write a CLIST similar to Figure 10 on page 74. The preceding 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('QMF810.SDSQEXIT',           +
               'QMF810.SDSLOAD',             +
                 'DSN;810.SDSNEXIT',           +
                 'DSN;810.SDSNLOAD')
/*****************************************************/
/*  Allocate  QMF libraries used for GDDM            */
/*****************************************************/
ALLOC FI(ADMGGMAP) DA('QMF810.SDSQMAPE')  SHR REUSE
ALLOC FI(ADMCFORM) DA('QMF810.DSQCFORM') SHR REUSE
ALLOC FI(DSQUCFRM) DA('QMF810.DSQUCFRM') SHR REUSE
ALLOC FI(ADMGDF)   DA('QMF72.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('QMF810.DSQPNLE') SHR REUSE
ALLOC FI(DSQDEBUG) SYSOUT(Z) LRECL(121) RECFM(F B A) BLKSIZE(1210)
ALLOC FI(DSQUDUMP) SYSOUT(Z) LRECL(125) RECFM(V B A) BLKSIZE(1632)
ALLOC FI(DSQSPILL) NEW UNIT(SYSDA) SPACE(1,1) CYLINDERS
ALLOC FI(DSQEDIT) NEW UNIT(SYSDA)
/*****************************************************/
/*  Issue ISPF LIBDEF for QMF libraries used for ISPF */
/*****************************************************/
ISPEXEC LIBDEF ISPLLIB LIBRARY ID(DSQLLIB)
ISPFEXE LIBDEF ISPPLIB DATASET ID('QMF810.SDSQPLBE')
ISPFEXE LIBDEF ISPSLIB DATASET ID('QMF810.SDSQSLBE')
ISPFEXE LIBDEF ISPMLIB DATASET ID('QMF810.SDSQMLBE')
/*****************************************************/
/*  Start QMF dialog using PASSLIB                   */
/*****************************************************/
ISPEXEC SELECT PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB
/*****************************************************/
/*  Free ISPF LIBDEF for QMF libraries used for ISPF */
/*****************************************************/
ISPEXEC LIBDEF ISPLLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPPLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPSLIB LIBRARY ID( )
ISPEXEC LIBDEF ISPMLIB LIBRARY ID( )
FREE       FI(DSQLLIB)
/*****************************************************/
/*  Free  QMF product data sets                      */
/*****************************************************/
FREE  FI(DSQPRINT)
FREE  FI(DSQPNLE)
FREE  FI(DSQDEBUG)
FREE  FI(DSQUDUMP)
FREE  FI(DSQSPILL)
FREE  FI(DSQEDIT)
/*****************************************************/
/*  Free  QMF libraries used for GDDM                */
/*****************************************************/
FREE  FI(ADMGGMAP)
FREE  FI(ADMCFORM)
FREE  FI(DSQUCFRM)
FREE  FI(ADMGDF)
```

Starting QMF in batch mode in ISPF

You can start QMF in batch mode to potentially save resources and time.

You can start QMF using ISPF with or without using a CLIST on z/OS. Place either of the following statements in the SYSTSIN data set of your JCL:

- Without a CLIST:

  ```
  ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(...DSQSMODE=B,DSQRUN=aa.bbb)
  ```

- With a CLIST:

  ```
  ISPSTART CMD(clist_name) NEWAPPL
  ```

  where *clist_name* is the name of the CLIST that starts QMF

PARM establishes the appropriate operating mode (DSQSMODE=B), identifies the procedure to be run (DSQSRUN=aaa.bb), and can include variables for that procedure.

The ellipsis after PARM represents optional parameter values that the user might want to include in addition to the required values for the DSQSMODE and DSQSRUN parameters. The name of the procedure must contain the authorization ID of the owner. For example, assume that a procedure was named PROCA and owned by the user authorization ID, JONES.

```
ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSMODE=B,DSQSRUN=JONES.PROCA)
```

Figure 11. Starting QMF in batch mode in ISPF with the user and procedure names

After the procedure runs, QMF ends and returns control to ISPF. ISPF can then continue with another procedure or command. When ISPF stops on z/OS, TSO runs the next TSO command in SYSTSIN. When all commands in SYSTSIN have been run, the job step ends.

Examples of starting QMF under ISPF: The following examples show how to start and pass parameters to QMF under ISPF:

- Starting ISPF from a CLIST (z/OS) and specifying QMF as the initial dialog:

  ```
  ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSIROW=150,DSQSRSTG=0)
  ```

  This statement passes a value of 150 for DSQSIROW (number of rows fetched before first display of report), and passes a value of 0 for DSQSRTG (amount of reserved storage).

- Starting from a CLIST that operates within ISPF on z/OS:

  ```
  ISPEXEC SELECT PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSSUBS=DB2SSFDX)
  ```

This statement passes the name DB2SSFDX for the DB2 subsystem.

- Starting from an ISPF menu:

```
)PROC

  &SEL = TRANS( TRUNC (&OPT,'.')
                1,'PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSPILL=NO)'
                .
                .
                .
```

This code passes NO for DSQSPILL.

- Starting from an CLIST and specifying an initial procedure:

```
ISPSTART PGM(DSQQMFE) 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.
  - To 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
    ```

Setting up and starting QMF on CICS

After QMF is tailored for CICS, start the QMF transaction (the default transaction is QMFE) from the CICS screen as follows:

```
QMFE parameters
```

where QMFE is the transaction ID, and parameters represents the desired program parameters.

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 terminal must be available. Also, make sure the ID is defined as either a local or a remote terminal on the system where the START command is issued.

Examples of starting QMF under CICS

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 screen display), passes a value of 500,000 for DSQSBSTG (maximum storage for reports), and turns off the 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
- To 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
```

Verify QMF data sets on z/OS

The following list of data sets is used by QMF in TSO. These files are allocated to DD names beginning with DSQ. If you want to allocate them differently, you must change the invocation exec.

Table 22. Data sets used by QMF in TSO

Data set	Purpose
DSQPNLE	QMF panel file
DSQUDUMP	QMF snap dump output
DSQDEBUG	QMF trace dump output
DSQSPRINT	Print data output
DSQSPILL	Spill data file
DSQEDIT	Edit transfer file

Verify program load libraries on z/OS

The DB2 database and the load libraries for ISPF, ISPF/PDF, QMF, DB2, and GDDM must be available from the STEPLIB statement or through a CLIST before starting QMF. Figure 12 on page 78 lists the load libraries.

```
//*************************************************************
//*     PROGRAM LOAD LIBRARIES                                *
//*************************************************************
//STEPLIB DD DSN=QMF810.SDSQLOAD,DISP=SHR     *     QMF MODULES  *
//  DD DSN=ISR.V41IM0.ISRLOAD,DISP=SHR  * PDF MODULES  * OPT.
//                                        for non-ISPF users
//  DD DSN=ISP.V4R1M0.ISPLOAD,DISP=SHR  * ISPF MODULES * OPT.
//                                        for non-ISPF users
//        DD DSN=DSN810.SDSNEXIT,DISP=SHR     * DB2 MODULES  *
//        DD DSN=DSN810.SDSNLOAD,DISP=SHR     * DB2 MODULES  *
//        DD DSN=GDDM230.SADMMOD,DISP=SHR     * GDDM MODULES *
```

Figure 12. Program load libraries for ISPF, ISPF/PDF, QMF, DB2, and GDDM

Verify GDDM data sets on z/OS: The GDDM data sets are allocated to the following DD names:

ADMGGMAP
GDDM map group for QMF-mapped panels

ADMCFORM
QMF-supplied chart forms

DSQUCFRM
User-defined ICUFORMS

```
//*****************************************************************
//*        QMF/GDDM DATA SETS                                     *
//*****************************************************************
//ADMGGMAP DD DSN=QMF810.SDSQMAPE,DISP=SHR   * GDDM Map Group
//ADMCFORM DD DSN=QMF810.SDSQCHRT,DISP=SHR  * QMF-Supplied Chart Forms
//DSQUCFRM DD DSN=aaaaaa,DISP=SHR           * Saves User-defined ICUFORMS
//ADMCDATA DD DSN=xxxx,DISP=SHR
//ADMGDF   DD DSN=xxxx,DISP=SHR
//ADMSYMBL DD DSN=xxxx,DISP=SHR
```

Figure 13. QMF/GDDM data sets

Chapter 10. Customizing your start procedure

This chapter describes the different ways that you can pass parameters to the program to customize a QMF session.

For information about passing parameters in a callable interface or in a REXX exec, see *Developing DB2 QMF Applications*.

Choosing virtual storage amounts for each session

QMF consists of several load modules. The main module (about 2.8 MB) can run in 31-bit mode above 16 MB and can be located in the extended pageable link pack area (EPLPA). A small support module (about 52 KB) must be run in 24-bit mode below 16 MB. This module can reside in the pageable link pack area (PLPA). By using EPLPA and PLPA, each z/OS region executing QMF can share QMF programs.

Each QMF region requires at least 1.5 MB of virtual storage. Additional storage generally provides improved performance, because QMF can keep more data records in virtual storage.

Program parameters for z/OS

When a user performs a QMF task that retrieves data from the database, the data is returned in a default report that is stored in virtual storage. This section explains QMF program parameters that help you customize:

- The maximum amount of storage used for report data
- The amount of spill storage used when virtual storage for reports is full
- The number of rows of data retrieved before QMF displays the first screen of the report

DSQSBSTG (adjusting storage for report data)

Parameter name
DSQSBSTG

Short form
B

Valid values
From 0 to 99,999,999 bytes

Default
0 bytes

The value of DSQSBSTG provides QMF with an upper limit (in bytes) on the storage available for report generation. It is a positive whole number ranging in value from 0 through 99, 999, 999. If DSQSBSTG is specified with a nonzero

value less than a QMF-determined minimum (15 to 32 KB, depending on the environment), it is increased to that minimum.

When DSQSBSTG has a value of 0, this parameter is not used; instead, DSQSRSTG is used to specify storage. However, if both DSQSBSTG and DSQSRSTG are specified, DSQSBSTG is used. The default for TSO, ISPF, or native z/OS is 0.

TSO performance tradeoffs

Use the DSQSPILL parameter to provide users with a spill file, which is the virtual I/O (UNIT=SYSVIO), or other DASD storage. If the spill file is full, QMF continues to retrieve data into virtual storage in amounts specified by the DSQSBSTG or DSQSRSTG parameters. The user does not receive any notification if there is insufficient storage, and QMF can complete its report processing. If you do not provide enough space, performance might be poor even when using a spill file, because QMF must return to the database several times to retrieve all the requested data. Users must make sure they have enough virtual storage for the QMF work they need to do.

Consider using a governor exit routine to limit rows retrieved from the database, so that less virtual storage is used for queries and reports. For more information about governor exit routines, see Chapter 19, “Controlling QMF resources using a governor exit routine,” on page 263.

CICS performance tradeoffs

Use the DSQSPILL parameter to provide users with a spill file. If the spill file is full, the QMF transaction is suspended until there is enough storage to satisfy the request.

Consider using a governor exit routine to limit rows retrieved from the database, so that less virtual storage is used for queries and reports.

DSQSRSTG (adjusting reserved storage used for applications)

Parameter name
DSQSRSTG
Short form
R
Valid values
From 0 to 99,999,999 bytes
Default
0

You can use the DSQSBSTG parameter if you want a more explicit specification of your report storage. The value of this parameter is a positive whole number ranging in value from 0 through 99,999, 999, with a default of 0. The value can affect other programs and the generation of reports.

The first time a user generates a report during a session, QMF determines how much storage is available in the QMF address space. The method that is used to arrive at the total storage acquired for QMF reports depends on both DSQSBSTG and DSQSRSTG:

- If DSQSBSTG is not specified, or is specified as 0, QMF subtracts the amount of DSQSRSTG from the total available storage to determine the maximum amount to use for QMF reports. The remaining storage is available for other programs, including z/OS system services, TSO commands, REXX, ISPF, and any other non-QMF user requirements.
- If DSQSBSTG is specified, then its value is used to determine how much storage is acquired for QMF reports, and DSQSRSTG is not used.

DSQSRSTG value of 0

You can specify 0 as the value for both DSQSBSTG and DSQSRSTG. In this case, the DSQSRSTG parameter is used and no storage is reserved for other system services. This value is probably adequate for users who never use z/OS, TSO commands, REXX, ISPF or other non-QMF services during QMF sessions. Those users who do use a z/OS system service or a TSO or command and has DSQSTSTG=0 and DSQSBSTG=0, run the risk of failing and causing an abend, because QMF does not reserve any storage for those services. Even the most casual users might unknowingly use a non-QMF program when they issue installation-defined QMF commands. Such commands are performed by QMF applications, which generally make extensive use of such non-QMF programs. Take this into account when selecting values for DSQSRSTG and DSQSBSTG.

Small value for DSQSBSTG or large value for DSQSRSTG

Requesting minimal storage for report processing can adversely affect performance when a user is handling a report. If enough storage is not available for the corresponding DATA object, QMF must use a spill file for excess rows of DATA. The input/output operations required for the spill file usually degrade performance.

DSQSPILL (acquiring extra storage)

Parameter name
DSQSPILL
Short form
L
Valid values
YES or NO
Default
YES

Because large amounts of report data in storage might affect the operation of other programs, QMF lets you allocate a spill file.

A spill file can improve performance in an interactive QMF session. Buffers in memory can store data so that QMF does not need to return to the database for multiple copies of the same data. Data the user needs to view multiple times does not need to be retrieved from the database several times; the spill file can be used to store it.

The spill file is activated automatically unless you specify NO:

```
DSQQMFn L=NO
```

Data is written to the spill file until:

- You use the RESET DATA command to reset the data object
- You replace the data object by running another query
- Your query has finished (all rows requested have been retrieved) and the data object is complete
- Your defined storage for the spill file (DFHTEMP in CICS, DSQSPILL is full

Allocating a spill file for non-CICS users

You can allocate a spill file through a FILEDEF statement or through a DD statement in the user's logon procedure, JCL, or CLIST. An example of this appears in the sample procedure, where the spill file is allocated through the DD statement, DSQSPILL. The statement looks like this:

```
//DSQSPILL  DD DSN=&&SPILL,DISP=(NEW,DELETE),
//             UNIT=SYSVIO,SPACE=(TRK,(1,9),RLSE),
//             DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
```

The statement:

- Allocates the spill file as a temporary data set, unique to the user's session
- Allocates the spill file to virtual I/O (UNIT=SYSVIO). You can allocate the spill file to other DASD storage instead.
- Specifies the DSQSPILL file with 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.)

The statement's SPACE operand can minimize spill file storage requirements during a session:

- The small primary extent keeps the space held by the spill file to a single track 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. This happens, for example, when the user begins a new report.

To allocate a spill file in a CLIST, use the following example:

```
ATTR SPILL RECFM(F) LRECL(4096) BLKSIZE(4096)
ALLOC FILE(DSQSPILL) UNIT(SYSVIO) SPACE(1,19) RELEASE +
  NEW DELETE USING(SPILL)
```

If the user waits to do this until a report is being generated, the spill file is not used for that report. The spill file is used during the session only when the underlying DATA object has been replaced (for example, through a DISPLAY command).

Estimating the space required for a spill file

If the data written to the spill file goes over the set limit (becoming full or unusable), QMF does not use the data from the spill file, but instead retrieves it again from the database, using virtual storage to hold it. You can exceed TSO DASD storage.

To accommodate QMF's storage requirements, ensure the TSO DASD storage is large enough to hold the individual spill files for all concurrent QMF users, in addition to any other transaction requirements for auxiliary temporary storage.

Use the following procedure to calculate the amount of space required for an individual spill file. Enlarge DFHTEMP storage according to how many individual spill files you will need to accommodate all concurrent users of QMF.

1. **Calculate the width (W) of one row of the largest table that can appear in the data object** by adding field widths in bytes (use Table 23 on page 84).
   - All rows of an individual table are the same width, regardless of the data each row contains. A row cannot be wider than 32,768 bytes.
   - Defined columns do not get written to the spill file.
2. **If W is 4,096 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 4,096 or less, QMF fits as many rows as it can into a page, without spanning pages.
3. **If W is greater than 4,096**, 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 4,096, 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.
4. **Calculate the number of pages required for the spill file**, according to the value of W:
   - If W is 4,096 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 4,096, calculate the number of pages required for the spill file by multiplying the number of rows in the table by P.

Table 23. Lengths of types of fields (use to estimate spill file size)

Field type	Field length in bytes
CHAR(n)	n+2
DATE	12
DECIMAL(n,m)	(n+1)/2+2, n odd (n+2)/2+2, n even
FLOAT(21)	10
FLOAT(53)	10
GRAPHIC(n)	n*2+2
INTEGER	6
SMALLINT	4
TIME	10
TIMESTAMP	28
VARCHAR(n)	n+4
LONG VARCHAR	(depends on other field lengths)
LONG VARGRAPHIC	(depends on other field lengths)
VARGRAPHIC(n)	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.

Table 24 shows a sample calculation for a spill file.

Table 24. Sample row width calculation for a spill file

Content of row	Calculation	Contribution to width
Two SMALLINT columns	2 x 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
Total width of row		59 bytes

The following sample calculations provide two ways to calculate the spill file space.

When R=4096/540 = 7 multiple rows/buffer:

```
600,000 rows    1 track    1 cylinder
------------ * --------- * ---------- = 571 cylinders
     7         10 blocks   15 tracks
```

When R=6000, 2 buffers/row:

```
6000 rows * 2 blocks/row *  1 track     1 cylinder
                            --------- *  ---------- = 800 cylinders
                            10 blocks    15 tracks
```

Using a spill file in a noninteractive QMF session

A spill file is most useful for improving performance in an interactive QMF session, when the DSQSMODE parameter is set to I. If you are running QMF noninteractively (the DSQSMODE parameter is set to B), using a spill file can also improve performance when multiple passes of the data are required to produce the report. A spill file might also be necessary to complete the data object, as when a RUN QUERY command is followed by a SAVE DATA command.

Multiple passes of the data are required when:

- You need to 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, the QMF program parameter DSQSPILL(YES/NO) should be set based on the work to be done. If the job is producing a large data object for printing, then allocating a spill file can have a negative effect on performance. In most cases when running in batch, DSQSPILL=NO is the best choice.

DB2 QMF Reference explains each of the QMF forms used to format reports and provides examples of how to use the forms.

Solving some spill file problems

Creating a spill file for your users can solve the user's storage problem, but can cause other problems. You might encounter problems with DASD space or create problems for other users.

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 the current reference to a specific volume.

Creating spill file problems for others: Increasing the spill file's secondary allocation could solve a user's spill file problem, but in doing this, you might create spill file problems for others. If you need to increase the secondary allocation, consider moving the user's spill file to a volume not used for the spill files of others.

A user can unknowingly create spill file problems for others. For example, a user might scroll to the bottom of a large table 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 table was first displayed and the time it was replaced by another. In the interim, the user's spill file might unnecessarily hold space that others need.

Performance problems: 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.

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.

Part of the processing time is devoted to writing the data to DSQSPILL so that it can be fetched later.

Performance is affected by several factors:

- The value of DSQSIROW (initial number of rows to fetch). This primarily affects the initial display of the report only.
- 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.
- Whether or not if data is fetched from the database the second time when multiple passes are required (not all data fits in memory and DSQSPILL), or from memory and DSQSPILL, or just from virtual memory.
- Whether you are scrolling backward or forward. Successive FORWARD commands usually perform best. BACKWARD commands might require starting over at the start of the answer set. This depends on the amount of memory, how far back you want to scroll, and the complexity of the report.

For very large answer sets with little memory and insufficient DSQSPILL allocation, the entire answer set could be read from row 1 to the new current row, every time the BACKWARD command is used.

The best performance is attained when there is sufficient memory to hold all data and DSQSPILL is not used.

If you can get the complete answer set into virtual memory before the first display (DSQSIROW is large), the database locks will be released. You will be able to scroll around the displayed report faster. This also slows the display of the first report screen. Releasing the locks could also improve performance for other users.

DSQSIROW (controlling the number of report rows retrieved for display)

Parameter name
DSQSIROW

Short form
F

Valid values
Any number from 0 through 99,999,999

Default
Minimum of 100 rows retrieved prior to the first report screen

Use DSQSIROW to specify the maximum number of rows 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:

- Executing queries that use SQL SELECT statements
- Displaying a database table with the QMF DISPLAY command

To determine the proper value for this parameter, use step 1 of the algorithm in "Estimating the space required for a spill file" on page 83 to estimate the size of a block of rows for the largest table a user is likely to query. A block is the number of rows that fit into one 4,096-byte buffer.

After every block of rows is retrieved, QMF compares the total number of retrieved rows to the value of DSQSIROW to determine whether to display the first screen of data. For example, suppose a block in your installation is 62 rows long, 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 percent (%) usage codes and ACROSS reports, require that all the data be retrieved before QMF displays the first screen. QMF ignores the DSQSIROW value in these situations. See *DB2 QMF Reference* for more information about these formatting options.

Performance with small DSQSIROW values

If you use too small a value for DSQSIROW, QMF might not be able to complete the data object before the first screen of data is displayed. An incomplete data object causes share locks on the data, which can prevent other users from updating the data.

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 were 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 noninteractive 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.

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.

Performance with large DSQSIROW values

If you use too large a value for DSQSIROW, QMF might take a long time to display the first screen of data. If you set DSQSIROW higher than you set the DSQSBSTG parameter, QMF might display a message indicating that there is insufficient storage available to satisfy the user's request.

When storage for the region is full, QMF waits for virtual storage to become available to finish retrieving rows for the database. QMF stops retrieving rows or terminates when storage is full.

Tracing QMF activity at the start of a session

QMF provides a trace facility that helps track user activity and any errors that might occur during a user's session. The program parameters explained in this section help you control:

- The level of detail at which QMF activity is traced, including activity before the user's profile is established
- Where trace data is stored

DSQSDBUG (setting the level of trace detail)

Parameter name
DSQSDBUG

Short form
T

Valid values
ALL or NONE

Default
NONE (no trace data)

Use DSQSDBUG to specify the level of detail at which you want to trace QMF activity. If you specify NONE, no trace is performed unless you load a profile with a saved value of ALL. If you specify ALL, ALL overrides the profile values and remains at ALL.

The tracing 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 NONE, until the profile is loaded.

Set DSQSDBUG to ALL when you want 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:

```
DSQQMFn T=ALL
QMFn T=ALL
```

For CICS, when you use a value of ALL, make sure the type of storage queue you choose is large enough to hold the trace output.

When you set DSQSDBUG to NONE, the level of detail in the trace output depends on whether the QMF session is running interactively or noninteractively:

- In either an interactive or a noninteractive 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 noninteractive session, all messages and commands are traced at the most detailed level.

After QMF starts, you can turn tracing off by using the command `SET PROFILE (TRACE=NONE`. You can also set more specific levels of trace detail using this command, by replacing `NONE` with various values that represent different QMF functions. See "Using the QMF trace facility" on page 338 for more information.

DSQSDBQT (specifying the type of CICS storage for trace data)

Parameter name
DSQSDBQT
Short form
(no short form)
Valid values
TD or TS
Default
TD (transient data queue)

Use DSQSDBQT to indicate the type of CICS storage you want to use for trace data. Specify the value TS to use a CICS auxiliary temporary storage queue for tracing:

```
QMFn DSQSDBQT=TS
```

Use temporary storage (TS) for message-level tracing. For other types of tracing, such as ALL, consider using a transient data queue if you think the trace output might exceed 32,767 rows of data (the limit for CICS temporary storage queues).

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 to CICS before you use it for the first time.

For more information on specifying the amount of detail in the QMF trace and viewing trace data, see "Using the QMF trace facility" on page 338.

DSQSDBQN (specifying the name of the CICS storage for trace data)

Parameter name
DSQSDBQN
Short form
(no short form)
Valid values
Any name that follows CICS naming conventions for queues
Default
DSQD

DSQSDBQN specifies the name of the transient data or temporary storage queue that holds trace data. A transient data queue named DSQD is predefined for you in the CICS DCT.

If you specify transient data for DSQSDBQT and you want to name the queue something other than DSQD, define the queue in the CICS DCT if it is not yet available.

Ensure 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
```

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.

Summary of program parameters

The following table displays the long and short forms of parameters and their appropriate environments. Parameters that are only used in CICS do not have a short form.

Table 25. Program parameters

Long form	Short form	Environment	Description
DSQSBSTG	B	TSO,CICS	Maximum storage for reports
DSQSDBCS	K	TSO,CICS	DBCS support of non-DBCS device
DSQSDBNM	D	TSO,CICS	Name of initial database location
DSQSDBQN	—	CICS	Name of CICS resource to use for QMF trace
DSQSDBQT	—	CICS	Type of CICS resource to use for QMF trace
DSQSDBUG	T	TSO,CICS	Trace — ALL or NONE
DSQSIROW	F	TSO,CICS	Rows fetched from the database
DSQSMODE	M	TSO,CICS	Interactive or batch mode
DSQSPILL	L	TSO,CICS	Use of the spill file
DSQSPLAN	P	TSO, CICS	Name of QMF application plan
DSQSPRID	U	TSO	Profile key — TSOID or PRIMEID
DSQSRSTG	R	TSO	Amount of reserved storage
DSQSRUN	I	TSO,CICS	Name of QMF procedure to run
DSQSSPQN	—	CICS	Name of QMF spill file
DSQSSUBS	S	TSO, CICS	Name of DB2 subsystem

Chapter 11. The QMF session control facility

The session control facility provides a method for initializing a QMF session by executing a specific QMF procedure when QMF is started. The name of the QMF procedure is Q.SYSTEM_INI. With this facility, the Q.SYSTEM_INI procedure can run any QMF command or any stored query that the user is authorized to run, prior to the user seeing the QMF home screen.

Installing Q.SYSTEM_INI

Create and save the Q.SYSTEM_INI procedure into the database like any other QMF procedure. The procedure must be named SYSTEM_INI and be saved under the authorization ID of Q. This QMF procedure should be shared among all QMF users. You can make the procedure sharable by specifying the SAVE command option SHARE=YES. It is also a good idea to add a comment describing the procedure. For example:

```
SAVE PROC AS Q.SYSTEM_INI (SHARE=YES,COMMENT='QMF System
 Initialization Procedure')
```

In order to save the procedure under the authorization ID of Q, the user must be a QMF Administrator. A QMF Administrator would have the global variable DSQAO_QMFADM equal to 1.

When does the Q.SYSTEM_INI procedure run?

The Q.SYSTEM_INI procedure runs just before the QMF initial procedure specified by the DSQSRUN parameter and just after QMF has completed initialization. All of the QMF functions available to QMF procedures are also available for use by the Q.SYSTEM_INI procedure.

Using Q.SYSTEM_INI

Your QMF session procedure Q.SYSTEM_INI, can be as simple as setting some QMF global variables or profile values, or as complex as complete front end to QMF. Each user can have their own session procedure called from, but not replacing, Q.SYSTEM_INI.

Example shipped with QMF

The sample Q.SYSTEM_INI procedure provided with QMF makes SHARE=YES the default for all users.

When you specify the DSQSUSER parameter, QMF issues a CONNECT command to connect to the database. Thus, the rules for this parameter are the same as for the CONNECT command.

- The ID you supply for the DSQSUSER parameter must have DB2 CONNECT authority, or the QMF session will not start. Use the SQL GRANT statement to grant this authority:

  ```
  GRANT CONNECT TO userid IDENTIFIED BY password
  ```

- The DB2 authorization ID and password you supply for DSQSUSER must conform to the rules for the CONNECT command for DB2.
- The SQL authorization ID and password must both be in the DB2 system table SYSIBM.SYSUSERAUTH for DB2 UDB for z/OS.

```
--
--  QUERY               D S Q 0 B I N I
--   MANAGEMENT         ---------------
--    FACILITY
--
--  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
```

Note: The actual example shipped with QMF may vary from the above example.

Figure 14. The Q.SYSTEM_INI shipped with QMF

User session procedure example

The session procedure can call another procedure. The procedure being called can be a user procedure that is created, owned and updated by a QMF user. You can use the same named procedure for different users if each user has a unique SQLID. When each user starts QMF they are running under their own SQLID. That SQLID is the default object owner when the object owner is not otherwise specified when accessing a QMF object or database object. For example, the QMF session procedure Q.SYSTEM_INI, could set global variables or company wide global variables and then call a user session procedure. In the following example, the user session procedure is called USER_INI.

```
PROC                Q.SYSTEM_INI                      LINE    1

-- This QMF procedure example shows how to setup QMF session defaults for
-- every QMF user and then calls a user procedure called USER_INI that will set
-- individual QMF session defaults
--
   QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1)   --  Process English Commands
   QMF RESET PROC                         --  Hide Contents of this PROC
   QMF SET PROFILE (WIDTH=80,LENGTH=66)   --  Set Default Report Page Size
   QMF SET PROFILE (SPACE=COMMON)         --  Set Default Space for Save Data Command
   QMF SET GLOBAL (DSQDC_LIST_ORDER=5D)   --  Object List Sorted by Date Modify
   QMF SET GLOBAL (DSQEC_RESET_RPT=1)     --  Prompt for Report Completion
   RUN USER_INI                           --  Run Users Session Procedure
   QMF END                                --  Display QMF Home screen first
   QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)   --  Return to Presiding Language
```

Figure 15. Q.SYSTEM_INI example that calls a user defined procedure

```
PROC        WILLIAMS.USER_INI                      LINE
1
-- This QMF procedure example shows how to setup 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 PROC
   QMF SET PROFILE (SPACE=MYSPACE)       --  Store data in MYSPACE.
   QMF SET PROFILE (PRINTER=MYROOM)      --  Print reports at My Printer
   QMF SET GLOBAL (DSQDC_LIST_ORDER=3A)  --  Object List Sorted by Object Name
   QMF SET GLOBAL (DSQEC_RESET_RPT=2)    --  Always ResetReports
   QMF SET GLOBAL (DSQEC_SHARE = 1)      --  Always Share My QMF Objects
   QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)  --  Return to Presiding Language
```

Figure 16. User session procedure example: user.USER_INI

Procedure that displays an object list

The following is an example of a SYSTEM_INI procedure that displays a list of objects instead of the QMF Home screen:

```
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
-- screen.
--
   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)   --  Object List sorted by object name
   QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)   --  Return to Presiding Language
   QMF LIST ALL                           --  LIST OBJECTS FOR ENGLISH
```

Figure 17. Using Q.SYSTEM_INI to display a list of objects rather than the QMF Home screen

Security and sharing session procedure

The QMF session procedure Q.SYSTEM_INI and other objects used or called by this procedure take on the same security as any other QMF object or database object does during a QMF session. The Q.SYSTEM_INI procedure is not special, other than QMF tries to execute it each time a QMF session is started. If the procedure does not exist, QMF does not attempt to run it.

If the Q.SYSTEM_INI procedure exists but is restricted or not shared, the result is the same as with any other QMF procedure object. If the SQLID starting QMF is Q, the procedure can run. Any SQLID other than Q receives a message that it is not authorized to run the procedure Q.SYSTEM_INI.

Diagnosis considerations

The QMF session procedure Q.SYSTEM_INI is run in the same environment as any other QMF procedure. All of the diagnosis procedures used for existing QMF procedures can also be used for the Q.SYSTEM_INI procedure. In addition to normal procedure execution, consider that this procedure is run before the QMF startup procedure named in the DSQSRUN parameter when QMF is started. If you have session controls in the procedure specified by the DSQSRUN parameter, consider moving them to the Q.SYSTEM_INI procedure.

You can use the QMF L2 tracing option to see commands and messages issued. Session procedure commands and messages are distinguished from others. See “Using the QMF trace facility” on page 338 for more information on QMF trace options.

Importing the default system initialization procedure on z/OS

A default QMF system initialization procedure is shipped on z/OS. The procedure is called DSQ0BINI. It can be found in QMF810.SDSQSAPE(DSQ0BINI).

You may want to verify if your system has a system initialization procedure before installing the sample The command DISPLAY Q.SYSTEM_INI will show you what is already installed, or issue the message, ″Q.SYSTEM_INI cannot be found″ if the initialization procedure has not been installed. If you already have a system intitialization procedure and wish to overwrite it with the sample, or do not have one and wish to install the sample, continue with the example below:

```
IMPORT PROC FROM'QMF810.SDSQSAPE(DSQ0BINI)'
```

You can import your own version of the procedure, import the default procedure, and change it before saving it. Or, you can create your own procedure from within QMF.

Chapter 12. QMF installation user exit (DSQUOPTS)

DSQUOPTS, a new QMF installation user exit for QMF Version 8.1, can be used to override the initial default value of selected global variables.

The global variables that are supported in the first level of DSQUOPTS are DSQEC_DISABLEADM and DSQEC_SHARE. Either or both of these global variables may have their initial default value set to a different value than the provided QMF default.

For example, DSQEC_DISABLEADM has a QMF initial default value of 0. This means that QMF will do QMF Administrator authority checking. If DSQUOPTS is modified to give DSQEC_DISABLEADM an initial value of 1, then QMF Administrator authority checking would not be done, and users that run QMF would never be considered to be QMF Administrators.

The QMF installation user exit DSQUOPTS may be modified by changing the DSQUOPTS assembler source, assembling and link-editing the module.

z/OS

For z/OS, the DSQUOPTS assembler source resides in member DSQUOPTS in the QMF810.SDSQUSRE data set. For details on how to specify override values, see the DSQUOPTS prolog. A sample job to assemble and link-edit DSQUOPTS resides in member DSQ1UOPT in the QMF810.SDSQSAPE data set. Note that the modified DSQUOPTS load module will be placed in the QMF exit library QMF810.SDSQEXIT. Remember to have the exit library properly allocated to pick up the modified exit. A default version of DSQUOPTS is shipped in the QMF810.SDSQLOAD data set.

Chapter 13. Establishing QMF support for end users

You can use QMF facilities to help customize support for end users. This chapter discusses how to set up QMF so that end users can access QMF and work with data in the database.

Creating user profiles to enable user access on TSO/CICS

Code page considerations: QMF receives information from and presents information to the terminal screen through services provided by GDDM. To prepare GDDM device support, specify the code page to use with QMF, or tailor GDDM session defaults, see the GDDM System Customization and Administration.

The role of Q.AUTHID: QMF installation automatically grants SYSADM authority to the user ID Q. The user Q owns and manages these QMF resources:

- All QMF control tables.
- The sample queries.
- The sample tables shipped with QMF. (For descriptions of the sample tables, see *DB2 QMF Reference* .)
- Default views for the database object list, explained in "Activating the enhanced object list" on page 125.

For the discussions and procedures throughout this book, we assume you are administering QMF using the Q user ID or another ID with SYSADM authority.

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

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. "Reading the Q.PROFILES table" on page 103 shows the Q.PROFILES table in detail and discusses possible profile values.

Establishing a profile structure for your installation

Provide users with a profile using one of these methods:

- 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 predefined in this row. The defaults used by this SYSTEM profile are discussed in "Adding a new user profile to the Q.PROFILES table." You can change these values to create a generic profile that meets the needs of your site.

- Create a unique row in Q.PROFILES for the use. 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 installation and allow other users to use the SYSTEM default profile; you can also delete the SYSTEM profile for security and tracking reasons, thus preventing those who do not have unique profiles from using QMF.

Adding a new user profile to the Q.PROFILES table

You can use SQL INSERT queries or the QMF Table Editor (described in *Using DB2 QMF*) to add new user profiles to the Q.PROFILES table. Figure 18 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). Use the TRANSLATION column of Q.PROFILES to distinguish between an English and an NLF environment.

The values shown in Figure 18 are examples of profile values you can use.

```
Base QMF (English)

INSERT INTO Q.PROFILES
(CREATOR, LANGUAGE, SPACE, TRANSLATION,
PFKEYS, SYNONYMS, RESOURCE_GROUP,
ENVIRONMENT)
VALUES ('JONES', 'PROMPTED', 'SAVEIT'
'ENGLISH', 'PFKEYS', 'COMMAND_SYNONYMS'
'NONPRIME', 'TSO')
```

```
German NLF

    INSERT INTO Q.PROFILES
    (CREATOR, LANGUAGE, SPACE, TRANSLATION,
     PFKEYS, SYNONYMS, RESOURCE_GROUP,
    ENVIRONMENT)
    VALUES ('SCHMIDT', 'MENUE', 'STUT2BER'
     'DEUTSCH', 'DEUTASTEN'
   'COMMAND_SYNONYM_D', 'SCHICHT'
    'TSO')
```

Figure 18. Creating a user profile on TSO

Note: 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. Figure 18 shows only a subset of all possible profile values. Use "Reading the Q.PROFILES table" on page 103 for guidance in specifying additional values.

To enroll several users, set up a template query that describes a standard profile and that uses a substitution variable value for any value that commonly changes (such as the value for the CREATOR column) with each new user you enroll. For more information on using substitution variables, see *DB2 QMF Reference*.

If you are using an NLF: You can establish different profiles for the same user according to the national language environment. 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.

Preventing users without unique profiles from using QMF

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. Figure 19 shows SQL statements that delete the rows. You can also use the Table Editor, as explained in *Using DB2 QMF*.

Base QMF (English)
German NLF

```
DELETE FROM Q.PROFILES
        DELETE FROM Q.PROFILES
WHERE CREATOR='SYSTEM'
        WHERE CREATOR='SYSTEM'
AND TRANSLATION='ENGLISH'
        AND TRANSLATION='DEUTSCH'
```

Figure 19. Restricting use of QMF to users who have unique profiles

Note: For both base QMF 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.

After you delete the SYSTEM row of Q.PROFILES, create a unique profile for every QMF user; otherwise, your users will not be able to use QMF.

Reading the Q.PROFILES table

Table 26 on page 104 shows the columns of the Q.PROFILES control table. Each column of the table represents an aspect of a user's QMF session you can customize. The defaults shown are for the English QMF environment.

If you are using an NLF: 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 version of the Q.PROFILE table. (Replace the n symbol with an NLID from Table 1 on page ix.)

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.

Table 26. Structure of the Q.PROFILES table

Column name	Data type and length	Nulls allowed	Function and possible values for z/OS
CREATOR	CHAR(8) when running in Compatibility mode. VARCHAR(128) when running in New Function Mode	No	**Function:** Specifies the authorization ID (the user) who owns the profile. **Values:** SYSTEM (default), primary or SQL authorization ID, or TSO logon ID, if DSQSPRID is set to TSOID. The SYSTEM row is shipped with Q.PROFILES for English and each NLF; users who do not have unique profile rows can use the SYSTEM row.
CASE	CHAR (18)	Yes	**Function:** Specifies whether terminal input is converted to uppercase. **Values:** UPPER (default), STRING, or MIXED. See *DB2 QMF Reference* for descriptions of these values. CASE might have a different default for NLF users.
DECOPT	CHAR (18)	Yes	**Function:**Specifies what separators QMF puts in numeric report columns. **Values:**PERIOD (default), COMMA, and FRENCH. See *DB2 QMF Reference* for more information. DECOPT is translated and might have a different default for NLF users.
CONFIRM	CHAR (18)	Yes	**Function:** Controls display of confirmation panels. **Values:** YES (default) if you want confirmation panels displayed before database changes; NO if you do not.

Table 26. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for z/OS
WIDTH	CHAR (18)	Yes	**Function:** Controls number of printed columns per page. **Values:** 22 to 999. Default = 132.
LENGTH	CHAR (18)	Yes	**Function:** Controls number of printed lines per page. **Values:** 1 to 999, or CONT if you want no page breaks. Default = 60.
LANGUAGE	CHAR (18)	Yes	**Function:** Controls which query language QMF uses when creating a new query after a RESET QUERY command is issued. **Values:** SQL (default), QBE (for Query-by-Example), or PROMPTED (for Prompted Query).
SPACE	CHAR (50)	Yes	**Function:** Specifies a table space that holds tables created using SAVE DATA and IMPORT commands. In DB2 Parallel Edition, this value refers to a NODEGROUP name. However, QMF refers to it as a TABLE SPACE name. Operation is not affected. DB2 DataJoiner does not utilize table spaces and the value for the SPACE option is ignored in a DB2 DataJoiner context; operation continues as if a blank value were present. **Values:** Any valid table space name.
TRACE	CHAR (18)	Yes	**Function:** Controls the level of detail in trace output. **Values:** 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 varies depending on the value for DSQSMODE. For example, when DSQSMODE is B, the trace level is L2, otherwise it is NONE. Only the values ALL and NONE are translated in NLFs.

Table 26. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for z/OS
PRINTER	CHAR (8)	Yes	**Function:** Controls where printer output is routed. **Values:** 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 GDDM-defined printer. **Reminder:** 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: `FREE DDNAME(DSQPRINT)`
TRANSLATION	CHAR (18)	No	**Function:** Indicates English or NLF environment **Values:** English (default) or the name of an NLF. The right-hand column of Table 1 on page ix shows the translated names you need to use in this column.
PFKEYS	CHAR(31) when running in Compatibility mode. LONG VARCHAR(261) when running in New Function mode.	Yes	**Function:** Indicates the table or view (if any) where user's customized function key definitions are stored. **Values:** Any valid DB2 table or view name. If blank or null (default), QMF's default keys are used.
SYNONYMS	CHAR(31) when running in Compatibility mode. LONG VARCHAR(261) when running in New Function mode.	Yes	**Function:** Indicates the table or view (if any) where user's customized command definitions are stored. **Values:** Any valid DB2 table or view name. If blank or null (default), no customized definitions are used. For NLF users, the IBM-supplied table is named Q.COMMAND__SYNONYM__n, where n is the National Language ID.

Table 26. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for z/OS
RESOURCE__GROUP	CHAR (16)	Yes	**Function:** Controls how the governor exit routine limits user's resources or commands. **Values:** Any valid resource group name. If blank or null (default), QMF attempts to use the user's SQL authorization ID here, and the user's session is not governed (unless the authorization ID is a valid resource group name).
MODEL	CHAR (8)	Yes	**Function:** Specifies the model for data access. **Values:** Always use the value REL for this column, indicating relational data.
ENVIRONMENT	CHAR (8)	Yes	**Function:** Indicates the operating environment. **Values:** This value is TSO, CICS if you access the profile through z/OS.

Providing the correct profile for TSO/CICS

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. You need to add the correct values to the user's profile to ensure that QMF recognizes them and starts.

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

userid is the authorization ID of the user trying to log on to QMF. DB2 uses this ID to determine if the user is authorized to use the database.

Current operating environment is CICS, z/OS, or TSO when QMF is being started from CICS or TSO respectively.

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.

Updating user profiles

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 this command 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. For more information on the SET PROFILE command and its parameters, see *DB2 QMF Reference*.

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 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. The Table Editor is explained in *Using DB2 QMF*.

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.

Use an SQL UPDATE query similar to the one in Figure 20 on page 109 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 query for user JONES in base (English) QMF; on the right is the same query for user SCHMIDT in the German NLF.

```
Base QMF (English)
        German NLF
UPDATE Q.PROFILES
        UPDATE Q.PROFILES
SET SYNONYMS='COMMAND__SYNONYMS'
        SET SYNONYMS='GUMMOW.XYZ'
WHERE CREATOR='JONES' AND
        WHERE CREATOR='SCHMIDT' AND
TRANSLATION='ENGLISH'
        TRANSLATION='DEUTSCH'
```

Figure 20. Updating user profiles using UPDATE query on Q.PROFILES table

Note: When running UPDATE, DELETE, and INSERT queries on the Q.PROFILES table, always include the TRANSLATION column in the query; otherwise, QMF applies the changes you make in all language environments.

Updating the SYSTEM profile

You can change the default values provided in the SYSTEM row of Q.PROFILES. 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 system has two resource groups defined, named PRIME and NONPRIME. 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.

Deleting profiles from the Q.PROFILES table

Periodically, you might need to delete obsolete user profiles from the Q.PROFILES table. Delete a user profile from Q.PROFILES when you are sure that objects created by the primary authorization ID or the TSO logon ID in that profile have been either deleted or safely transferred to other users:

- For information on how to perform these tasks for QMF queries, forms, and procedures, see “Maintaining QMF objects using QMF control tables” on page 138.
- For instructions on database tables and views, see “Maintaining tables and views using DB2 tables” on page 148.

Use a query similar to the one shown in Figure 21 on page 110 to delete a user profile.

```
Base QMF (English)
        German NLF
DELETE FROM Q.PROFILES
        DELETE FROM Q.PROFILES
WHERE CREATOR='JONES'
        WHERE CREATOR='SCHMIDT'
AND TRANSLATION='ENGLISH'
        AND TRANSLATION='DEUTSCH'
```

Figure 21. Deleting a QMF user profile

If you are using an NLF: Include 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 TRANSLATION, QMF deletes the profile in all NLF environments.

Deleting profiles on z/OS

If the user whose profile you deleted had a private table space, use the SQL DROP TABLE SPACE 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. *DB2 UDB for z/OS SQL Reference* explains the DROP statement. *DB2 QMF Reference* explains the ERASE command.

Granting and revoking SQL privileges

Users automatically own any objects they create and save in the database (unless they create a table with a different owner). The owner of an object automatically has all SQL privileges on objects he or she owns, and can grant (or revoke) these privileges to other users. Anyone with DB2 administrator authority can grant or revoke SQL privileges for any object in the database. The user Q has this authority, and is predefined to DB2 during QMF installation.

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, and form names. In these cases, the user's SQL authorization ID serves as the implicit qualifier. For example, a user is operating with JONES as the current SQL authorization ID. During the session, the user issues the command:

```
RUN QUERYA (FORM=FORMA
```

which runs the following SQL query:

```
SELECT * FROM TABLEA
```

The RUN command refers to the query JONES.QUERYA and the form JONES.FORMA. The SELECT command refers to the table JONES.TABLEA.

If you 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 DBA authority to create a table, view, or index with a qualified name that is not your current authorization ID.

Using the SQL GRANT statement

Use the SQL GRANT statement to grant SQL SELECT, UPDATE, INSERT, and DELETE privileges. For example, suppose user JONES needs to issue the following command:

```
EDIT TABLE ORDER__BACKLOG (MODE=CHANGE
```

Assuming that you are the owner of the table, use the following statement to to grant JONES the SQL UPDATE privilege he needs to edit the ORDER__BACKLOG table in change mode:

```
GRANT UPDATE ON ORDER__BACKLOG TO JONES WITH GRANT OPTION
```

WITH GRANT OPTION indicates that JONES can grant to other users any of the SQL privileges you granted him for the ORDER__BACKLOG table.

If you need to run GRANT queries often, use QMF variables in place of parts of the query that frequently change, such as UPDATE, ORDER__BACKLOG, and JONES. Variables are explained in *DB2 QMF Reference*. You might also consider using a QMF procedure to do the task if there is more than one query. *Using DB2 QMF* explains how to create procedures.

Use the keyword PUBLIC to grant SQL privileges to all QMF users. For example, use the statement in Figure 22 to grant INSERT authority on the ORDER__BACKLOG table to all users, and allow each of those users to grant INSERT authority to other users:

```
GRANT INSERT ON ORDER__BACKLOG TO PUBLIC WITH GRANT OPTION
```

Figure 22. Granting an SQL privilege to all QMF users

For more information on the GRANT statement, see the appropriate *DB2 UDB SQL Reference*.

Note: If you grant more than one person INSERT, UPDATE, or DELETE privileges on a database object, and two or more users try to access that object at the same time, there might be contention for resources, causing performance or other problems. If a user is editing a table required during QMF initialization, that table can be locked to prevent QMF from starting for other users.

Using the SQL REVOKE statement

Use the SQL REVOKE statement to remove privileges:

```
REVOKE UPDATE ON ORDER__BACKLOG FROM JONES
```

Figure 23. Revoking an SQL privilege from a QMF user

Use the PUBLIC keyword to revoke privileges from all QMF users.

DB2 privileges have a cascading structure; privileges revoked from a user are automatically revoked from any additional users to whom that user granted them.

Controlling 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. Because this data might be sensitive, you may need to control users' access to certain objects:

- You can use SQL GRANT and REVOKE statements from QMF's SQL query panel to control access to tables and views.
- You can use the SHARE parameter of the QMF SAVE command to control access to queries, forms, and procedures.

Controlling access on z/OS

All QMF users need access to the QMF application plan and packages built by DB2 during QMF installation. The plan and packages enable QMF to run as a DB2 application program. At installation time, the QMF plan and packages are GRANTed EXECUTE to PUBLIC. You can REVOKE and issue specific GRANTs to the user IDs/groups if this is preferred.

Providing access to the application plan and packages

You can enable users to use QMF by granting the EXECUTE privilege to PUBLIC or to individual users with the SQL GRANT query.For example, to grant access to user JONES:

```
GRANT EXECUTE on plan QMF__PLAN
to JONES
```

If you provide access to the QMF plan and packages by individual, you must execute an SQL GRANT statement for each new user.

If you restrict access by individual user, you limit use of the plan and packages to selected DB2 primary or secondary authorization IDs. The difference in refinement shows up when two or more primary authorization IDs have use of the same secondary authorization ID. If you use restricted enrollment to QMF through the profile, then only the primary authorization IDs that have rows in Q.PROFILES have access to QMF. If you restrict access to QMF based on granting EXECUTE privilege to specific authorization IDs, then anyone who has these authorization IDs as their primary or secondary authorization IDs has access to QMF.

Revoking user access to the QMF application plan and packages

After you dispose of a user's queries, forms, and procedures, you need to remove the user's access to the QMF application plan and packages if you granted the access individually. You can run the following queries:

```
REVOKE EXECUTE on plan 'QMF__PLAN'
FROM 'JONES'

REVOKE EXECUTE on package 'QMF__PACKAGE'
FROM 'JONES'
```

Revoke the EXECUTE authority on all packages used by QMF.

If the user's EXECUTE privilege was granted more than once, you must revoke each grant individually using the following queries:

```
REVOKE EXECUTE on plan 'QMF__PLAN'
FROM 'JONES' by all

REVOKE EXECUTE on package 'QMF__PACKAGE'
FROM 'JONES' by all
```

You must have SYSADM authority on z/OS to revoke a GRANT.

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.

If other users share the same authorization ID of the former user, do not revoke access to the plan and packages from the authorization ID. If you do, the users sharing the authorization ID will no longer be able to use QMF.

DB2 privileges required to access objects

The DB2 privileges required to run QMF queries, the Table Editor, and QMF commands are the same privileges needed to run the underlying SQL statements.

Distributing DB2 privileges is a two-step process:
1. Assigning the user a set of authorization IDs
2. Assigning DB2 privileges to the authorization IDs

To assign and revoke privileges:
- Assign authorization IDs through a DB2 exit routine.
- Assign DB2 privileges through SQL GRANT queries.
- Undo previous grants through SQL REVOKE queries.

Not every query run in a QMF session requires DB2 privileges. Those that do not are called static queries and are in the QMF code. QMF uses these queries, for example, to update its own control tables. Users who have nothing to do with QMF administration do not need DB2 privileges on these tables.

The privilege to run dynamic queries comes exclusively from the user. Dynamic queries include all queries executed with the RUN command. They also include certain queries formulated on behalf of the user by QMF. For example, the user issues the DISPLAY command to see the contents of a table.

DB2 privileges required for QMF commands, for prompted and QBE queries, and for the Table Editor are the same as those listed for SQL in "SQL privileges required to access objects" on page 121.

Granting and revoking DB2 privileges

You provide DB2 privileges by running GRANT queries that give DB2 privileges to one or more authorization IDs. For example, the following query grants the SELECT and UPDATE privileges on the table SMITH.TABLEA to the authorization IDs JONES and JOHNSON:

```
GRANT SELECT, UPDATE ON TABLE SMITH.TABLEA TO JONES, JOHNSON
```

Run REVOKE queries to withdraw grants of DB2 privileges. You can always withdraw grants for which your SQL authorization ID is the grantor. For example, in a QMF session, the user's current SQL authorization ID is JONES. JONES had previously granted the SELECT privilege on the table SMITH.TABLEA to BAKER. The following query withdraws this grant of the privilege:

```
REVOKE SELECT ON TABLE SMITH.TABLEA FROM BAKER
```

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.

Granting to PUBLIC: You can make grants to PUBLIC and to individuals. Granting a privilege to PUBLIC makes it available to all local users.

To make an object available to remote and local users for DB2 UDB for z/OS subsystems that have distributed data enabled, grant authority to PUBLIC AT ALL LOCATIONS. For example, the following queries 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 sample tables of QMF. This, and similar queries for the other sample tables, are run during QMF installation, so that everyone has SELECT privilege on the sample tables.

Granting privileges to users: The privilege to run a GRANT query must come from the grantor; that is, from the user's current SQL authorization ID. The grantor must have every privilege being granted and must have each privilege with the GRANT option. For example, BAKER wants to grant the SELECT and UPDATE privileges on the table SMITH.TABLEA to JONES. To do this, BAKER must have the SELECT and UPDATE privileges with the GRANT option on the same table.

A GRANT query can include the expression WITH GRANT OPTION. When it does, the privileges are granted with the GRANT option. Without the GRANT option, users cannot grant authority to others. For example, the following queries grant the SELECT privilege on SMITH.TABLEA to JONES and JOHNSON. After the queries are run, only JOHNSON can grant the privilege to others.

```
GRANT SELECT ON TABLE SMITH.TABLEA TO JONES
GRANT SELECT ON TABLE SMITH.TABLEA TO JOHNSON WITH GRANT OPTION
```

You may have received your DB2 privilege through a SQL GRANT query, from SYSADM authority on z/OS, or because you own the created object. Any DB2 privilege you have might be the result of a chain of grants, beginning with a grant from someone with *installation SYSADM* authority. Installation SYSADM authority is the highest DB2 UDB for z/OS authority that anyone can have. During DB2 installation, one or two authorization IDs receive this authority. Users operating with this authority can then grant lesser privileges to others, to be granted in turn to others, and so on.

Granting specific privileges: To grant a specific privilege, one of your authorization IDs must have the privilege to do so, and this ID must be your current SQL authorization ID. If this ID is not your current SQL authorization ID, logon to that ID, or if possible, run the SET CURRENT SQLID query.

Granting table privileges: The most commonly used privileges for a table are SELECT, INSERT, UPDATE, and DELETE. When you grant the SELECT privilege on a table, the grantee can select data from it in a SELECT query or

subquery. When you grant the INSERT, UPDATE, or DELETE privilege on a table, the user can modify the table's data.

If you own a given table, you have all the table privileges with the GRANT option.

Granting view privileges: View access can be granted to screen-sensitive data, to read only, and to create.

Views as screening tools: You can use views in place of the tables they represent to screen sensitive data from the viewers. For example, you want to create a view 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 create such a view with the following query:

```
CREATE VIEW VIEWA AS
  SELECT NAME, DEPT, JOB, YRS
    FROM SMITH.STAFF
```

View owners and underlying objects: Granting a privilege for a view begins with the owner of the view. In this book, the owner of the view is assumed to be the creator. 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 query. For example, the underlying object of the view created with this query is the table SMITH.STAFF:

```
CREATE VIEW VIEWA AS
  SELECT NAME, DEPT, JOB, YRS
    FROM SMITH.STAFF
```

View privileges and read only views: The view privileges are SELECT, INSERT, UPDATE, and DELETE. 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 that the view represents.

The owner of a view has the SELECT privilege on the view, but might not have other privileges. The other privileges might be lacking if the owner of the view did not have the privilege on the underlying object. Alternatively, the privileges might be lacking because the view is read only.

A view is read only, if the defining query is a join. Queries other than joins can also appear in read only views. For more on read only views, see the description of CREATE VIEW queries in the *DB2 UDB for z/OS SQL Reference*.

Privilege to create a view: To create a view, the user's SQL authorization ID must have the SELECT privilege on each of the underlying objects of the view. No other privilege is needed.

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 using that view as an underlying object are also dropped, and so on.

Granting view privileges: A person with the GRANT option on some view privilege can grant that privilege to others using the GRANT option. The grantee needs no privilege at all on the underlying objects. This fact makes views useful for screening data: with no privilege on the underlying objects, users granted the SELECT privilege on a view can see only the view. If users need SELECT privilege on the underlying objects, they can bypass the view and query these objects directly.

Privileges of the owner of the view: The owner normally creates one or more tables, and then one or more views of these tables. For each of these views, the owner has SELECT privilege with the GRANT option. If a view is not read only, the owner also has INSERT, UPDATE, and DELETE privileges with the GRANT option. The owner can then grant these privileges to others.

Views with other types of underlying objects: The owner of both the tables and views has a complete set of privileges, with the GRANT option, on the underlying objects. 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 may 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.
  - The owner of the view has the same privilege on the single underlying object.

Authority to maintain a database on z/OS: Suppose that, after creating a database, you want someone else to maintain it. With proper DB2 authority, you can grant that user DBADM authority over the database. With this authority, the user can perform maintenance tasks, such as:

- Create and drop table spaces and tables from the database

- Create and drop indexes for the tables of the database
- Run utilities for maintaining the tables and indexes

The holder of this authority also has a full set of privileges on the database tables, no matter who actually owns them. For example, if you want authorization ID JONES to have the power to maintain the database DBASEA, run this query:

```
GRANT DBADM ON DATABASE DBASEA TO JONES
```

You can run this query if your SQL authorization ID has SYSADM authority or is the owner of the database.

DBADM authority on a database also has the CREATETS privilege, which lets you create table spaces for the database, and the CREATETAB privilege, which lets you create tables in the database.

If you can grant DBADM authority on a database, you can also grant lesser privileges. Moreover, anyone having DBADM authority with the GRANT option on the database can do the same. For example, if you want the authorization ID JONES to have the power to grant lesser privileges on database DBASEA, run this query:

```
GRANT DBADM ON DATABASE DBASEA TO JONES WITH GRANT OPTION
```

Granting the appropriate privilege: SAVE and IMPORT commands on z/OS: Use the IMPORT command sparingly in CICS, because it can affect the performance of other users in the same address space. Also, QMF uses OS QSAM services GET/PUT. This can lock out other QMF users in the same CICS region during I/O operations.

QMF must have the DB2 privileges to run the queries that result from the SAVE and IMPORT commands. The privilege must come from the user, as if the user were running the queries through the RUN command. For example, a user must have the INSERT privilege on a table or an authority implying the INSERT privilege before QMF can run the INSERT queries on that table.

Determining what privileges are needed*:* The privileges needed depend in part on whether the user is creating his or her own tables or tables for other users.

When users create tables for others, the qualifier (the owner of the object) must be the user's primary or secondary authorization ID. In creating a table for another user, other privileges might let the appropriate CREATE table query run, but might not let INSERT queries run.

When users create their own tables after the table structure is created, the users automatically have the necessary INSERT privilege. All that is needed is

the privilege to run the CREATE TABLE query. The minimum privilege to do this depends on which table space option was chosen:

Explicit option
The user needs at least CREATETAB privilege on the database and USE privilege on the receiving table space.

Implicit option
The user needs at least CREATETAB and CREATETS privilege on the database.

The user of the default DB2 UDB for z/OS database, DSNDB04, might already have some of these privileges. During DB2 installation, the CREATETAB and CREATETS privileges for the default database were 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.

Note: The database might be the DB2 UDB for z/OS default database (DSNDB04). However, it should not be one of the databases used exclusively by DB2 itself: DSNDB01, DSNDB03, or DSNDB05.

Granting the necessary privileges*:* Through one or more of the following queries, you can grant the privileges that your user lacks:

```
GRANT CREATETAB ON DATABASE &dbname TO &authid
GRANT CREATETS ON DATABASE &dbname TO &authid
GRANT USE OF TABLE SPACE &dbname.&tbspname TO &authid
```

where:
&dbname
Specifies the name of the database.
&authid
Specifies the user's authorization ID.
&tbspname
Specifies the name of the receiving table space.

Do not enclose these values in quotation marks. For example, if you want to grant USERA the CREATETAB privilege on the database DATABSE2, run this query:

```
GRANT CREATETAB ON DATABASE DATABASE2 TO USERA
```

You have the authority to run these queries if you have the privileges they grant, and you hold these privileges with the GRANT option. This is true if you have SYSADM or SYSCTRL (for DB2 2.3) authority or if you have DBADM, DBCTRL, or DBMAINT authorities with the GRANT option.

Revoking the grants of others on z/OS: If your SQL authorization ID has SYSADM authority, you can revoke the grants of others. This gives you a way of revoking privileges, even if they are a result from multiple grants. For example, BAKER has the SELECT privilege on the table SMITH.TABLEA. JONES wants to remove this privilege from BAKER, but does not know who the grantors are. JONES, who has SYSADM authority, has the authority to run the following query:

```
REVOKE SELECT ON TABLE SMITH.TABLEA FROM BAKER BY ALL
```

BY ALL removes every grant of the privilege.

Revoking a grant to PUBLIC on z/OS: You can withdraw a grant of privilege from PUBLIC, just as you can from a single authorization ID. Doing so, however, does not remove this privilege from those who gained the privilege from another source.

You cannot remove a table privilege from the owner of a table. Nor can you remove an implied database privilege, such as CREATETAB, from someone with, for example, DBADM authority over a database. For more on what you can and cannot do with a REVOKE query, see *DB2 UDB for z/OS Administration Guide*. Also, see the description of the REVOKE command in the *DB2 UDB for z/OS SQL Reference*.

What can happen when too many users can grant DB2 authority: Revoking a DB2 privilege might withdraw it from more users than you intended. This is known as the cascade effect, because some authorities depend on the existence of others. For example, a privilege held because of a single grant is lost if the grantor loses that privilege. BAKER has the SELECT privilege with the GRANT option on SMITH.TABLEA. BAKER grants this privilege to JOHNSON and JONES. For JOHNSON and JONES, this is the only source of this privilege. A REVOKE query now removes this privilege from BAKER. As a result, the query removes this privilege from JOHNSON and JONES.

The loss of privileges can spread to many users, especially if some of those who lost privileges had 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, this might be the SELECT privilege on a table. When this happens, no one can run the program.

Both the cascade effect and ineffective revoking of grants are more likely when many users have the ability to grant DB2 privileges.

SQL privileges required to access objects

Whenever a SELECT query is issued through QMF, either through one of the QMF query interfaces or as a result of commands, such as DISPLAY TABLE or PRINT TABLE, QMF adds FOR FETCH ONLY to the query to improve performance when accessing remote data. Therefore, FOR FETCH ONLY should not be added to SQL queries run through QMF.

SQL privileges required for QMF commands: Using Table 27, locate the QMF command your users need to use and grant them the required SQL privilege on the table or view they are working with.

Table 27. QMF commands and their SQL equivalents

This QMF command:	Requires this SQL privilege on objects referenced by the command:
DISPLAY table/view	SELECT
DRAW table/view	SELECT
EDIT TABLE table/view	The necessary privileges depend on the Table Editor mode.
EXPORT TABLE table/view	SELECT
IMPORT TABLE table/view	If the table exists, SELECT, DELETE, and INSERT. To include a comment, you must have either ownership of the table or DBADM authority for the table's database. If the table does not exist, you must have either the CREATETAB privilege or DBADM authority for the database or the USE privilege for the table space specified in the SPACE field of your user's profile.
PRINT table/view	SELECT
RUN query	Whatever privileges are used in the query
RUN procedure	Whatever privileges are used in the commands in the procedure
SAVE DATA	If the table exists, SELECT, DELETE, and INSERT. To include a comment, you must have either ownership of the table or DBADM authority for the table's database. If the table does not exist, you must have either the CREATETAB privilege or DBADM authority for the database or the USE privilege for the table space specified in the SPACE field of your user's profile.
LIST table/view	SELECT

Not all users can use the SAVE command to create a new table.

For more information on SQL privileges, such as SELECT, INSERT, UPDATE, or DELETE, see *DB2 UDB for z/OS SQL Reference*.

SQL privileges required for prompted and QBE queries: Using Table 28, 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 28. QMF query types and their SQL equivalents

Users using this type of query:	Need this SQL privilege:
PROMPTED	SELECT
QBE I.	INSERT
QBE P.	SELECT
QBE U.	UPDATE
QBE D.	DELETE

For more information on prompted or QBE queries, see *Using DB2 QMF*.

SQL privileges required for the table editor: Using Table 29, locate the Table Editor function your users need to use and grant them the SQL privilege on the table or view they need to edit.

Table 29. Table Editor commands and their SQL equivalents

Users using this Table Editor function:	Need this SQL privilege on tables or views being edited:
ADD	INSERT
SEARCH	SELECT
CHANGE	UPDATE
DELETE	DELETE

For more information on the Table Editor, see *Using QMF*.

Using the SQL GRANT statement

Use the SQL GRANT statement to grant SQL SELECT, UPDATE, INSERT, and DELETE privileges. For example, suppose user JONES needs to issue the following command:

```
EDIT TABLE ORDER__BACKLOG (MODE=CHANGE
```

Assuming you are the owner of the table, use the statement in to grant JONES the SQL UPDATE privilege he needs to edit the ORDER__BACKLOG table in change mode:

```
GRANT UPDATE ON ORDER__BACKLOG TO JONES WITH GRANT OPTION
```

Figure 24. Granting SQL privileges to a single QMF user

WITH GRANT OPTION indicates that JONES can grant to other users any of the SQL privileges you granted him for the ORDER__BACKLOG table.

If you need to run GRANT queries often, use QMF variables in place of parts of the query that frequently change, such as UPDATE, ORDER__BACKLOG, and JONES. Variables are explained in *DB2 QMF Reference*. You might also consider using a QMF procedure to do the task if there is more than one query. *Using QMF* explains how to create procedures.

Use the keyword PUBLIC to grant SQL privileges to all QMF users. For example, use the statement below to grant INSERT authority on the ORDER__BACKLOG table to all users, and allow each of those users to grant INSERT authority to other users:

```
GRANT INSERT ON ORDER__BACKLOG TO PUBLIC WITH GRANT OPTION
```

Figure 25. Granting an SQL privilege to all QMF users

For more information on the GRANT statement, see *DB2 UDB for z/OS SQL Reference*.

Note: If you grant more than one person INSERT, UPDATE, or DELETE privileges on a database object, and two or more users try to access that object at the same time, there might be contention for resources, causing performance or other problems. If a user is editing a table required during QMF initialization, that table can be locked to prevent QMF from starting for other users.

Sharing QMF objects with other users

You or any QMF user can enable access to QMF queries, forms, and procedures by using the SHARE parameter of the QMF SAVE command.

Specify SHARE=YES when saving 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 command below 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 26. Sharing a QMF object

The default is defined by the global variable DSQEC_SHARE. See *DB2 QMF Reference* for more information.

The owner of an object can change its shared status at any time, using a DISPLAY command followed by a SAVE command, as shown below:

```
DISPLAY ORDER__QUERY
SAVE QUERY AS ORDER__QUERY (SHARE=NO
```

Figure 27. Changing the shared status of a QMF object

For more information on the SAVE command, see *DB2 QMF Reference*.

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.

Uncommitted read can be useful in a distributed environment. However, allowing uncommitted read can introduce non-existent data into a QMF report. Do not allow uncommitted read if your QMF reports must be free of non-existent data.

Values can be:

'0' Isolation level UR, Uncommitted Read.

'1' Isolation level CS, Cursor Stability. This is the default.

For DB2 QMF Version 8.1 the use of the value '0' is only effective with the database server DB2 for OS/390 Version 4 or higher.

Setting standards for creating objects

The objects in your installation might be shared among many users, so they should have names that indicate what the object is and how it should be used. Encourage users to provide comments that describe for other users the purpose of queries, forms, procedures, and tables. Tables and views require more maintenance and administration, so consider establishing special guidelines for creating these objects.

For information on how to create comments for QMF and database objects using the SAVE command, see *DB2 QMF Reference*.

Activating the enhanced object list

The enhanced object list allows users to list DB2 tables that belong to group IDs, tables that are owned by the user, and tables available for public viewing. In this case, table privileges are granted to group, rather than user, IDs. Any user who can access these group IDs or secondary authorization IDs have the privileges.

You must install and activate a User Defined Function (UDF) supplied by QMF to use the enhanced object list. The UDF must be installed into a DB2 UDB for OS/390 V6 or later database. To install and activate the enhanced object list, do the following steps:

1. Set up the environment for UDFs. This 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. Read the DB2 UDB for z/OS documentation for more information on setting up the WLM-established address space for DB2 stored procedures and UDFs.
2. Add the QMF program DSQABA1E to the WLM-established address space that will execute the QMF-supplied UDF. DSQABA1E resides in the QMF load library QMF810.SDSQLOAD. Copy the DSQABA1E member from SDSQLOAD into a load library in the STEPLIB concantenation for the WLM-established stored procedure address. This is the STEPLIB concantenation defined in the JCL PROC used to start the address space.
3. Get the name of the WLM environment where the QMF-supplied UDF will run. This is specified by the APPLENV= parm of the JCL PROC used to start the WLM environment; it is required to complete the next step. When the DB2 installer specifies Option 6, WLM Environment, on the DB2 install panel DSNITIPX, DB2 assumes a default. This is recorded as ZPARM "WLMENV" in macro DSN6SYSP and is listed prominently in the DB2 Install Job Stream DSNTIJUZ.
4. Define the QMF UDF to DB2. This is a registration activity which is performed by running the QMF-supplied job DSQ1BUDF located in the QMF810.SDSQSAPE library. This job issues an SQL CREATE FUNCTION statement and grants execution privileges. You may need to tailor job DSQ1BUDF prior to running it.
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:

```
SELECT U.AUTHNAME FROM TABLE( Q.APPL_AUTHNAMES( 'PUBLIC "PUBLIC*"' )) U
```

The result should be a list of valid authorization names for the user who is executing the SQL statement above. Here is an example of how it might look:

```
AUTHNAME
--------
W397754
#DQZA
#J49A
DB2FUNC
QMFDEV
PUBLIC
PUBLIC*
```

6. Change the QMF list view to execute the QMF UDF. Run QMF-supplied job DSQ1BUDV located in the QMF810.SDSQSAPE to change the view.
7. (Optional) If you tailored the IBM-supplied tables view, read the following SELECT statement which has been modified to use the QMF-supplied UDF. This example will help you in modifying your customized view:

```
SELECT T.CREATOR, T.NAME,...
   FROM SYSIBM.SYSTABLES T
   ,( SELECT DISTINCT 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*"' )) U
            ) AS UAT ("CREATOR", "NAME")
              WHERE T.CREATOR=UAT.CREATOR AND
               T.NAME=UAT.NAME AND T.TYPE
               IN ('T', 'V')
```

Follow the job sequence in the table below to install an Enhanced List view into a QMF V8.1 Compatibility or New Function mode for DB2 UDB for z/OS server V8.1

Table 30. Job sequence to install Enhanced List View

Job name	Purpose
DSQ1BSQL	Binds the install programs to the current server
DSQ1BUDF	Creates the Enhanced QMF List View Function
DSQ1BUDV	Creates Enhanced QMF List Views

If the Enhanced List View does not function like you want it to, run job DSQ1BVW to restore the QMF default List Views.

QMF users periodically need to list objects they have saved in the database or to view comments that show them what purpose a table serves or what type of data a column in the table contains. The QMF LIST and DESCRIBE commands perform these functions.

When a user issues a LIST or DESCRIBE command for a table, QMF uses a view defined on a set of DB2 catalog tables to obtain information about the table. The name of this view is stored in the global variables DSQEC__TABS__LDB2, DSQEC__TABS__RDB2, or DSQEC_TABS_SQL. When users issue these commands for a column within a table, QMF uses the global variables DSQEC__COLS__LDB2, DSQEC__COLS__RDB2, or DSQEC_COLS_SQL to obtain the name of the view.

QMF provides a set of default views, loaded during installation, that return only the tables and column information the user is authorized to see. Because processing for authorization takes extra time and resources, QMF also allows you to customize the table lists and column information by creating your own views.

Using the default object lists

For a complete list of the views provided by QMF, refer to Appendix B. QMF provides the following default views and automatically assigns them to the user Q during installation into DB2 UDB for z/OS databases:

Q.DSQEC__TABS__LDB2
Q.DSQEC__TABS__RDB2
Q.DSQEC__COLS__LDB2
Q.DSQEC__COLS_RDB2
Q.DSQEC__ALIASES

QMF also provides SQL default views that you might need in a remote unit of work environment:

Q.DSQEC__TABS__SQL
Q.DSQEC__COLS__SQL

The view Q.DSQEC__TABS__LDB2 selects only the list of tables and views from the current location in DB2 UDB for z/OS, and workstation or iSeries database servers. Figure 28 on page 128 shows the view provided for DB2 UDB for z/OS.

```
CREATE VIEW Q.DSQEC__TABS__LDB2
    (OWNER,TNAME,TYPE,SUBTYPE,MODEL,RESTRICTED,REMARKS,
     CREATED,MODIFIED,LAST_USED,LABEL,LOCATION,OWNER__AT__LOCATION,
     NAME__AT__LOCATION)
  AS SELECT DISTINCT
    CREATOR,NAME,'TABLE',TYPE,' ',' ',REMARKS,' ',' ',' ',
    LABEL,LOCATION,TBCREATOR,TBNAME
    FROM SYSIBM.SYSTABLES, SYSIBM.SYSTABAUTH
    WHERE CREATOR = TCREATOR AND NAME=TTNAME AND GRANTEETYPE = ' ' AND
         GRANTEE IN (USER,'PUBLIC',CURRENT SQLID,'PUBLIC*')
```

Figure 28. Default view that provides a list of tables for the LIST command (z/OS)

To use a view you have created (for example, QMFADM.LOCAL__DB2__TABLES) and override the default view, issue a command like this one:

```
SET GLOBAL (DSQEC__TABS__LDB2 = QMFADM.LOCAL__DB2__TABLES
```

The view Q.DSQEC__TABS__RDB2 selects only the list of tables and views in a remote DB2 location accessed through a three-part name or the LOCATION option of LIST. The user's current location must be DB2 UDB for z/OS.

```
CREATE VIEW Q.DSQEC__TABS__RDB2
    (OWNER,TNAME,TYPE,SUBTYPE,MODEL,RESTRICTED,REMARKS,
     CREATED,MODIFIED,LAST__USED,LABEL,LOCATION,OWNER__AT__LOCATION,
     NAME__AT__LOCATION)
  AS SELECT DISTINCT
    CREATOR,NAME,'TABLE',TYPE,' ',' ',REMARKS,' ',' ',' ',
    LABEL,LOCATION,TBCREATOR,TBNAME
    FROM SYSIBM.SYSTABLES, SYSIBM.SYSTABAUTH
    WHERE CREATOR = TCREATOR AND NAME=TTNAME AND GRANTEETYPE = ' ' AND
         GRANTEE IN (USER,CURRENT SQLID,'PUBLIC*')
```

Figure 29. Default view that provides a list of tables for the LIST command (z/OS)

To use a view you have created (for example, QMFADM.REMOTE__DB2__TABLES) and override the default view, issue a command like this one:

```
SET GLOBAL (DSQEC__TABS__LDB2 = QMFADM.REMOTE__DB2__TABLES
```

If you are a remote user: You do not have access to objects defined only as PUBLIC at the relevant remote location.

The view Q.DSQEC__ALIASES selects only the list of aliases for a list of tables, or the column information for an alias in DB2 UDB for z/OS, DB2 workstation, or iSeries servers.

```
CREATE VIEW Q.DSQEC__ALIASES
    (OWNER,TNAME,TYPE,SUBTYPE,MODEL,RESTRICTED,REMARKS,
     CREATED,MODIFIED,LAST__USED,LABEL,LOCATION,OWNER__AT__LOCATION,
     NAME__AT__LOCATION)
  AS SELECT
    CREATOR,NAME,'TABLE',TYPE,' ',' ',REMARKS,' ',' ',' ',
    LABEL,LOCATION,TBCREATOR,TBNAME
    FROM SYSIBM.SYSTABLES
    WHERE CREATOR IN (USER,CURRENT SQLID) AND TYPE = 'A'
```

Figure 30. Default view that provides a list of aliases for the LIST command (z/OS)

To use a view you have created (for example, QMFADM.DB2__ALIASES) and override the default view, issue a command like this one:

```
SET GLOBAL (DSQEC__ALIASES = QMFADM.DB2__ALIASES
```

```
CREATE VIEW Q.DSQEC__COLS__LDB2
    (OWNER, TNAME, CNAME, REMARKS,LABEL)
  AS SELECT DISTINCT
    TBCREATOR, TBNAME, NAME, REMARKS, LABEL
  FROM SYSIBM.SYSCOLUMNS, SYSIBM.SYSTABAUTH
    WHERE TCREATOR = TBCREATOR AND TTNAME = TBNAME AND GRANTEETYPE = ' '
      AND GRANTEE IN (USER,'PUBLIC',CURRENT SQLID,'PUBLIC*')
```

Figure 31. Default view that provides column information for the DESCRIBE command (z/OS)

To use a view you have created (for example, QMFADM.LOCAL__DB2__COLUMNS) and override the default view, issue a command like this one:

```
SET GLOBAL (DSQEC__COLS__LDB2 = QMFADM.LOCAL__DB2__COLUMNS
```

To use a view you have created (for example, QMFADM.LOCAL__DB2__COLUMNS) and override the default view, issue a command like this one:

```
SET GLOBAL (DSQEC__COLS__LDB2 = QMFADM.LOCAL__DB2__COLUMNS
```

The view Q.DSQEC__COLS__RDB2 selects only the column information from a table on another DB2 location. The user's current location must be DB2.

To use a view you have created (for example, QMFADM.REMOTE__DB2__COLUMNS) and override the default view, issue a command like this one:

```
SET GLOBAL (DSQEC__COLS__RDB2 = QMFADM.REMOTE__DB2__COLUMNS
```

If you are a remote user: You do not have access to objects defined only as PUBLIC at the relevant remote location.

The views shipped with QMF 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.

Changing the default list

Using the QMF-provided default views for your table lists and column information might increase processing time, because DB2 gathers authorization information from the SYSIBM.SYSTABAUTH. 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.

Use a query similar to the one in Figure 32 to create your own view. This query 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. See *DB2 QMF Reference* for information on QMF naming conventions.

```
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 32. Customizing your object lists using global variables

Remember to SET GLOBAL for the appropriate variable for the new view name to be used.

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 list. 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.

Follow these rules if you're creating a list view of your own:

- The view must have the same view column names as the corresponding QMF-supplied view. 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.
- Do not exceed the following maximum lengths for columns in the view:
  - 18 characters for TNAME, CNAME, and NAME__AT__LOCATION
  - 254 characters for REMARKS
  - 30 characters for LABEL
  - 1 character for RESTRICTED
  - 16 characters for LOCATION
  - 8 characters for OWNER, TYPE, SUBTYPE, MODEL, and OWNER__AT__LOCATION
- Always supply values for OWNER, TNAME, TYPE, and CNAME. These columns cannot be null.

DSQEC__TABS__LDB2, DSQEC__TABS__RDB2, DSQEC__ALIASES, DSQEC__COLS__LDB2, and DSQEC__COLS__RDB2 are part of a set of global variables that help you control aspects of a user's QMF session. For more information on using global variables in procedures, see *Using DB2 QMF*. For a list of global variables and information on using them in applications, see *Developing DB2 QMF Applications*.

Object list storage requirement

For the LIST command, there are two sets of storage requirements for each row of the object list.

- 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, and FORM, 63 bytes

- The storage to hold 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.

Note: For a complete list of the views provided by QMF refer to Appendix B, "QMF objects residing in DB2," on page 363.

Enabling users to create tables in the database

A QMF user can create a table using any of these methods:

- SQL CREATE TABLE statement

  Enter the SQL CREATE TABLE statement from a QMF SQL query panel or run it from a saved query.

- QMF DISPLAY TABLE (or DISPLAY *viewname*) command, followed by the SAVE DATA command

  All SQL privileges on the underlying table or view are required. If the name you specify on the SAVE DATA command is the name of an existing table, QMF replaces or appends the existing data object. The SAVE command might be rejected if the table attributes do not match. For more information on the SAVE DATA command, see *DB2 QMF Reference* and the online help.

- QMF IMPORT TABLE or IMPORT VIEW command

  All SQL privileges on the table or view being imported are required. If the name the user specifies on the IMPORT command is the name of a table that already exists, QMF replaces or appends the data in the existing table. The IMPORT command might be rejected if the table attributes do not match. For more information on the IMPORT command, see *DB2 QMF Reference* and the online help.

Depending on the needs of your installation, you might need to create tables for your users or enable them to create their own tables.

Creating tables on z/OS

Table 31. Creating tables in the database

If you are creating tables for your users:	If users are creating tables themselves:
Step 1 Create a table space and define it to DB2 before its first DB2 use. Use the appropriate *DB2 UDB Administration Guide* to help you decide on assigning authorities to create table spaces or dbspaces.	**Step 1** Use the *DB2 UDB for z/OS Administration Guide* to grant a user DB2 CREATETS authority or DB2 CREATETAB authority. Create a table space (if you have only given them CREATETAB authority) and define it to DB2 before its first use.

Table 31. Creating tables in the database (continued)

If you are creating tables for your users:	If users are creating tables themselves:
Step 2 To create the table, issue either an SQL CREATE TABLE statement, a QMF DISPLAY command followed by a SAVE DATA command, *or* an IMPORT TABLE command. See *Using DB2 QMF* for examples of creating tables.	**Step 2** 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.
Step 3 Create one or more indexes on the tables you create, to improve DB2 performance. See the *DB2 UDB for z/OS SQL Reference* for information on the CREATE INDEX statement and details on logical design of tables.	**Step 3** 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.
Step 4 Fill the tables with data. Use the DB2 UDB for z/OS LOAD Utility, QMF IMPORT commands (for transferring small tables), or other methods. *DB2 UDB for z/OS Utility Guide and Reference* explains how to use the LOAD Utility. *Using DB2 QMF* explains exporting and importing objects in QMF.	**Step 4** Provide education on the SQL CREATE TABLE statement, QMF SAVE DATA and IMPORT commands, and other guidelines your site has for creating tables. See *DB2 QMF Reference* for more information on these commands.
Step 5 Grant DB2 and SQL privileges for the tables to users who need them.	**Step 5** 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.

For more information on the CREATE TABLE, CREATE INDEX, and other SQL statements related to creating tables, see *DB2 UDB for z/OS SQL Reference*.

Choosing and assigning a table space for the user

A table space can be either assigned to or created by the user. Any QMF user with CREATETAB authority can create tables in an assigned table space. If the table space is owned, only the owner can create tables in it unless they assign authority to others. For additional guidance on table spaces, see *DB2 UDB for z/OS Administration Guide*.

When creating a table space, you must choose between the two options: explicit and implicit.

Explicit

With this option, all the tables created by the user's SAVE and IMPORT commands appear in a single table space created with an SQL CREATE TABLESPACE command. In DB2 terminology, this table space is "explicitly created". For example,

```
UPDATE Q.PROFILES
  SET SPACE='DBASE1.TSPACE1'
  WHERE CREATOR='USERA' AND TRANSLATION='ENGLISH'
```

Implicit

With this option, each table created by the user's SAVE and IMPORT commands goes into a table space created exclusively for that table by DB2. In DB2 terminology, this table space is "implicitly created". Such table spaces have the default LOCKSIZE, BUFFERPOOL, STOGROUP, and space attributes, and have names derived from their table names. For example,

```
UPDATE Q.PROFILES
  SET SPACE='DATABASE DBACE1'
  WHERE CREATOR='USERA' AND TRANSLATION='ENGLISH'
```

For information on the default attributes, see the description of the CREATE TABLESPACE query in *DB2 for z/OS SQL Reference*.

For information on the table spaces, see *DB2 UDB for z/OS Administration Guide*.

You need to consider the following factors when you decide between the options for the table space.

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 command can run successfully. Procedures to do this are described in the *DB2 for OS/390 and z/OS Administration Guide*.

Maintenance

When you use the QMF Explicit Table Space Option, you simplify maintenance if you take advantage of segmented table spaces. Implicitly created table spaces can also simplify maintenance.

For example, if the user creates various temporary tables and then erases them, creating and erasing these tables in a simple table space (not segmented) causes a rapid buildup of dead space that would

soon have to be removed by reorganizing the table space. In contrast, 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. An implicitly created table space is erased automatically when the table it contains is erased.

Resource contention

To avoid resource contention, use either the explicit table space option with a segmented table space, or the implicit table space option.

With a segmented table space, when a table is locked, the lock does not interfere with access to segments of other tables. Having a number of tables in a single simple table space, each used by more than one user, might cause resource contention, but placing the tables in a segmented or separate table space might avoid the resource contention.

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 database. 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 is already available for user created tables. It is created during QMF installation and used for the installation verification procedure. The table space is named DSQTSDEF, and its database is DSQDBDEF. You might find that this table space is large enough to hold the tables of your users.

Many users should use this table space only if the tables are primarily read only.

Choosing the type of table space: You can choose from three types of table spaces for your users.

Simple
Segmented
Partitioned

For more information about the types of table spaces, see the *DB2 UDB for z/OS Administration Guide*.

Granting a user DB2 CREATETAB authority (z/OS)

You need to grant DB2 CREATETAB authority to any user who needs to create tables in a database. To grant a user CREATETAB authority, issue the

SQL statement shown in Figure 33, where *userid1*, *userid2*, and *userid3* represent SQL authorization IDs.

```
GRANT CREATETAB on database DBASEA TO userid1, userid2, userid3, ...
```

Figure 33. SQL statements to grant CREATETAB authority to more than one user

A user with CREATETAB authority can create tables in a table space. Users with CREATETS authority can create a table space for their own use.

If you want to allow a user to create tables, but need 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.

See the *DB2 UDB for z/OS Administration Guide* for more information on creating a table space and a discussion of DB2 authority levels.

Enabling users to support a chart

QMF creates charts using the Interactive Chart Utility (ICU) supplied by the GDDM-PGF product. Chart formats are templates for various types of charts (such as pie charts or histograms) that don't contain data. When a user creates a chart, QMF associates the data used with the chart format. Then, 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.

Supporting a chart in TSO and ISPF

From a single report, users can specify different chart forms, such as scatter charts, pie charts, and bar charts. Users can use IBM-supplied chart forms or create their own. In addition, they can save newly created chart forms, if they have libraries in which to store them.

To create a library to hold a user's saved chart forms:

1. Create the new library with a DD statement like this:

```
//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, using the ddname DSQUCFRM. You might allocate the data set through the user's TSO logon procedure, or you might allocate it through a CLIST that the user calls to reach QMF. For example:

   ```
   ALLOC DSNAME(aaaaaaaa) DDNAME(DSQUCFRM) SHR
   ```

The IBM-supplied chart forms are in the library QMF810.SDSQCHRT. Allocate this library to the DD name ADMCFORM. Both this library and the user's library are searched for user specified chart forms, but the new library is searched first. When the user saves a chart form, it always goes into the new library, never into QMF810.SDSQCHRT.

This arrangement gives each user access to both the IBM-supplied chart forms and those the user saved. It also prevents replacement of the IBM-supplied chart forms.

Supporting a chart in CICS on z/OS

QMF users can create charts from their reports through the interactive chart utility (ICU), a feature of GDDM. From a single report, users can specify different chart forms, such as scatter charts, pie charts, and bar charts. Users can use IBM-supplied chart forms or create their own. In addition, they can save newly created chart forms, provided they have libraries in which to store them.

During QMF installation, a data set is created to hold IBM-supplied charts. This data set is described to CICS by an FCT or CSD file entry with the name DSQUCFRM. This data set is normally allocated to the CICS region during CICS start up and is available to all CICS users. The DSQUCFRM data set is the default chart library used to store chart forms when using the ICS from QMF. You can store chart forms into other chart libraries by using the advanced form of the ICU panel directory. Each chart library must be described to CICS and accessed by the CICS region that is executing QMF. You describe the chart library with an FCT or file entry in the CSD data set. For a description on how to use the advanced ICU panel directory, see the*GDDM Presentation Graphics Feature Interactive Chart Utility User's Guide*.

In addition to the ICU, 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 into the GDDM ADMF library. You can also save the whole chart in GDF using the ICU facility of GDDM.

Maintaining QMF objects using QMF control tables

Periodically, you need to condense and reorganize the QMF control tables that store QMF queries, forms, and procedures. Regular maintenance of the QMF control tables 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.

For a complete list of QMF Control Tables provided by IBM, please refer to Appendix B. QMF Objects Residing in DB2.

All QMF queries, forms, and procedures are stored among three QMF control tables:

- The Q.OBJECT__DIRECTORY table, which is described in "Reading the Q.OBJECT__DIRECTORY table"
- The Q.OBJECT__DATA table, which is described in "Reading the Q.OBJECT__DATA table" on page 139
- The Q.OBJECT__REMARKS table, which is described in "Reading the Q.OBJECT_REMARKS table" on page 140

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 need to assign STATS and REORG privileges to a user who is monitoring or reorganizing the control tables.

Reading the Q.OBJECT__DIRECTORY table

This table contains a row for each QMF query, form, and procedure in the database. The table has the index Q.OBJECT__DIRECTORYX, with attributes UNIQUE and CLUSTER. The keyed columns are OWNER and NAME. No two rows can have identical values for these columns.

The Q.OBJECT__DIRECTORY table has the structure shown in Table 32:

Table 32. Structure of the Q.OBJECT__DIRECTORY table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	CHAR	8	No	Shows the authorization ID of the creator of the object.
NAME	VARCHAR	18	No	Shows the name of the object.
TYPE	CHAR	8	No	Shows the type of object: FORM, PROC, or QUERY.

Table 32. Structure of the Q.OBJECT__DIRECTORY table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
SUBTYPE	CHAR	8	Yes	Shows SQL, QBE, or PROMPTED when TYPE is QUERY. Null or blank if TYPE is not QUERY.
OBJECTLEVEL	INTEGER	4	No	QMF uses this number to reconstruct an object from its defining text in the Q.OBJECT__DATA table.
RESTRICTED	CHAR	1	No	YES if the object has not been shared (using the SHARE parameter of the QMF SAVE command); 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	Shows the date value for when an object was last used. The value is updated only once a day.

Reading the Q.OBJECT__DATA table

This table contains one or more rows for each query, form, and procedure in the database. Each row contains all or part of the defining text for one of these objects. 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__OBJDATAX, with attributes UNIQUE and CLUSTER. Keyed columns are OWNER, NAME, and SEQ.

The table has the structure shown in Table 33 on page 140:

Table 33. Structure of the Q.OBJECT__DATA table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	VARCHAR	Determined by database	No	Shows the authorization ID of the creator of the object.
NAME	VARCHAR	Determined by database	No	Shows the name of the object.
TYPE	CHAR	8	No	Shows the type of object: FORM, PROC, or QUERY.
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	LONG VARCHAR FOR BIT DATA (see note)	Determined by database	Yes	Contains all or part of text that defines the object. Text appears in an internal QMF format. The OBJECTLEVEL column in Q.OBJECT__DIRECTORY defines this format. **Attention:** The APPLDATA column must never be subjected to code page (CCSID) conversion.

Reading the Q.OBJECT_REMARKS table

This table contains one row for each query, form, and procedure in the database. The row contains comments entered using the QMF SAVE command when the object was created or last replaced. (See the description of the SAVE command in *DB2 QMF Reference*.)

The Q.OBJECT__REMARKS table has the index Q.OBJECT_REMARKSX, with the attributes UNIQUE and CLUSTER. Keyed columns are OWNER and NAME.

The table has the structure shown in Table 34:

Table 34. Structure of the Q.OBJECT__REMARKS table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	CHAR	Determined by database	No	Shows the authorization ID of the user who created the object

Table 34. Structure of the Q.OBJECT__REMARKS table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
NAME	VARCHAR	Determined by database	No	Shows the name of the object.
TYPE	CHAR	8	No	Shows the type of the object: FORM, PROC, or QUERY.
REMARKS	VARCHAR	Determined by database	Yes	Contains the comment that was saved with the object when it was created or replaced.

Listing QMF queries, forms, and procedures

To get the information you need to help you maintain the QMF environment, you need to list the queries, forms, and procedures that QMF users have saved in the database. With administrator authority you can list QMF objects you do not own using the query in Figure 34.

```
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 forms, and procedures owned by a particular user

This query returns a list of objects sorted by type (FORM, PROC, QUERY) and further by subtype (SQL, QBE, or PROMPTED) if TYPE is query. Enclose the value you supply for `userid` in single quotation marks. Objects of each type are further sorted by whether they've been shared by the owner. Shared status is reflected in the RESTRICTED column of the Q.OBJECT__DIRECTORY table.

Displaying QMF queries, forms, and procedures

If listing the objects does not provide enough information in the REMARKS column, try displaying the object by one of the following methods:

- Running the following query to share the user's objects, then displaying them from your own ID:

```
UPDATE Q.OBJECT_DIRECTORY
    SET RESTRICTED = 'N'
    WHERE OWNER = 'userid'
```

Figure 35. Sharing another user's objects with all users

Enclose the value you supply for `userid` in single quotes.

Note: Run this query only if you do not need to track which of the user's objects are restricted and which are not. After you run this query, you can reset RESTRICTED to Y, but then you will not be able to tell which objects were originally restricted.

- Issuing the QMF DISPLAY command for each object you want to display.

Transferring ownership of queries, forms, and procedures

Use the queries shown in Figure 36 to transfer QMF objects from one user to another. Make sure that you run all three queries.

Note: First make sure that the new owner has no objects saved with the name of the object you are transferring, or QMF will replace the existing object with the object that you transfer.

```
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
```

Figure 36. Transferring QMF objects to another user

In the queries shown in Figure 36, `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 query like the one in Figure 35 to change the RESTRICTED column value to Y if you decide you want to share the object after transferring it.

Deleting obsolete queries, forms, and procedures

Use the SQL in Figure 37 to delete all of a particular user's QMF queries, forms, and procedures. Make sure you run all three queries, because the internal representation of each object spans the three QMF control tables Q.OBJECT__DIRECTORY, Q.OBJECT__DATA, and Q.OBJECT__REMARKS. Surround values you supply for the user ID variables with single quotes.

Unpredictable results can occur if the tables are not properly updated.

```
DELETE FROM Q.OBJECT_DIRECTORY    DELETE FROM Q.OBJECT_REMARKS      DELETE FROM Q.OBJECT_DATA
WHERE OWNER = 'olduserid'         WHERE OWNER = 'olduserid'         WHERE OWNER = 'olduserid'
```

Figure 37. Deleting unnecessary objects from the QMF control tables

You can also delete obsolete objects by using the date and time sorting capabilities in Q.OBJECT_DIRECTORY. You can select every object where the date last used was before 06/01/95 and delete all the appropriate rows from the three control tables.

Importing queries, forms, and procedures in z/OS data sets

If a user has QMF objects that have been exported to z/OS data sets, you can bring them back with the QMF IMPORT command.

If the exported objects are RACF protected, you need RACF read access to import objects from them. To obtain this access, see your RACF administrator.

Enlarging the table space for the QMF object control tables

Periodically, QMF objects might become too numerous for the table space that contains the QMF object control tables Q.OBJECT_DIRECTORY, Q.OBJECT_DATA, and Q.OBJECT_REMARKS.

Before you enlarge the table space, you must determine its space requirements. One factor in your estimate can be the amount of space currently used.

If the space is DB2 managed, you can get this information by doing the following:

1. Run the STOSPACE utility on the table space's storage group.
2. Run the following query:

```
SELECT SPACE
  FROM SYSIBM.SYSTABLEPART
  WHERE TSNAME='tttttttt' AND DBNAME='DSQDBCTL'
```

   where `tttttttt` is the table space name. The result (SPACE) gives the number of kilobytes of storage currently allocated to the table space.

If the space is user managed, you can use the TSO LISTCAT command for the space information, if you know the data set name.

To enlarge the table space for the QMF object control tables:

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 should produce 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 DB2 managed.

Table 35. Table spaces for control tables that store 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	5120 pages

Table 36. Node groups for control tables that store QMF objects using a DB2 Parallel Edition V1R2 database or DB2 Universal Database

NODEGROUP Name	Used for	Characteristics
DSQTSCTL	For all QMF control tables except as described elsewhere in this table.	Can be distributed across multiple nodes. Growth potential is low.
DSQTSOBJ	The QMF OBJECT control tables where PROC, Query, and FORM objects are stored.	Can be distributed across multiple nodes. 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 node to avoid complications.

Table 36. Node groups for control tables that store QMF objects using a DB2 Parallel Edition V1R2 database or DB2 Universal Database (continued)

NODEGROUP Name	Used for	Characteristics
DSQTSAMP	The QMF Sample tables.	Candidate for distributing across multiple nodes.

3. Stop the database.

   Use the command `-STOP DATABASE(DSQDBCTL)`.
4. Change the table space description.
   - If the table space data sets are user managed, issue a DB2 statement of the following form:

     ```
     ALTER TABLESPACE DSQDBCTL.tttttt
       USING STOGROUP ssssss PRIQTY pppp SECQTY ssss
     ```

     where `tttttt` is the table space name. The statement changes the table space from user managed to DB2 managed 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 DB2 managed, execute a DB2 statement like the following:

     ```
     ALTER TABLESPACE DSQDBCTL.tttttt
       PRIQTY pppp SECQTY ssss
     ```

     where `tttttt` is the table space name. `pppp` and `ssss` are 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 effect enlargement. You must instead do something that causes the table space to be refilled.
6. Start the database with the statement:

   ```
   -START DATABASE(DSQDBCTL)
   ```

You can also use the DB2 LOAD utility to enlarge a table space.

For more information on enlarging table spaces, see *DB2 UDB for z/OS Utility Guide and Reference*.

Note: DB2 QMF Version 8.1 creates DB2 managed table space data sets if QMF was not previously installed.

Maintaining a DB2 subsystem on z/OS

Note: Except where noted, this section provides information about DB2 UDB for z/OS.

You can maintain multiple databases with multiple table spaces.

Workstation database server users: Each server (a named location) is a single database. You can maintain multiple table spaces in that single database.

You might assign specialized administration tasks to users to perform under their own authorization IDs. Give these users just enough DB2 authority to run the queries and utilities required for their tasks. For example, a person would need:

- The INSERT privilege on the table Q.PROFILES to insert QMF profiles for new users
- DBADM authority on a given database to administer the associated tables, indexes, and table spaces
- STATS and REORG privileges on the database for the Q.OBJECT tables to monitor these tables and, if necessary, reorganize them

Managing data sets

The data sets for the table spaces and indexes might be user or DB2 managed. How these data sets are managed determines what you must do to enlarge table spaces and indexes.

Storage groups for DB2 managed data sets

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, as Table 37 indicates.

Workstation database server users: Storage groups do not apply.

Table 37. Control table storage groups

Table	Table space	Storage group
Q.PROFILES	DSQTSPRO	DSQSGPRO
Q.ERROR__LOG	DSQTSLOG	DSQSGLOG
Q.OBJECT__DIRECTORY	DSQTSCT1	DSQSGCT1
Q.OBJECT_REMARKS	DSQTSCT2	DSQSGCT2
Q.OBJECT__DATA	DSQTSCT3	DSQSGCT3

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 queries. 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.

Maintaining the control tables

Most control-table maintenance cannot be done under QMF, because QMF relies on these tables for its operations. You can issue your maintenance queries in batch-mode TSO through the DSN processor, or interactively through the SPUFI facility of DB2I.

Workstation database server users: Additionally, you can use the DB2 Command Line Processor from the local operating system environment of the database.

You can find information on these subjects in the *DB2 UDB for z/OS Administration Guide*.

No one should be using QMF during maintenance work. To ensure this, apply the DB2-STOP DATABASE command to one of the table spaces containing a control table. You can then do maintenance operations on the other control tables and indexes. You can do either of the following:

- Include the DB2-STOP DATABASE command as the first in your input to DSN if you are working in batch-mode TSO.
- Issue the DB2-STOP DATABASE command from the DB2I commands panel if you are using DB2I.

For a description of the DB2-STOP DATABASE command, see *DB2 UDB for z/OS Utility Guide and Reference*.

Monitoring and reorganizing the control tables

You should forestall maintenance problems by monitoring the condition of the control tables through the DB2 system catalog. For more information, see the *DB2 UDB for z/OS Administration Guide*.

Running the RUNSTATS utility: You periodically run the RUNSTATS utility on the control tables and indexes to add current statistics to certain DB2 system tables. You then query these tables and examine these statistics to decide whether reorganization is required.

If reorganization is required, do the following:

1. Run the REORG utility.
2. Rerun the RUNSTATS utility.
3. Query the updated system tables again to see if the reorganization improved the statistics.

At its most effective, reorganization can minimize the space requirements for the control tables and indexes and increase the efficiency of QMF operations.

The *DB2 UDB for z/OS Administration Guide* suggests that you rebind your most critical applications after reorganization so that the most efficient search paths can be selected. This suggests that the QMF application plan be rebound after each such reorganization.

Switching buffer pools

For performance reasons, you might want to change the buffer pool for a table space containing a control table or for a control table index. For example, if your installation is strongly QMF oriented, you might 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 queries. For descriptions of these queries and the authorities needed to run them, see *DB2 UDB for z/OS SQL Reference*. You can choose BP0, BP1, or BP2 for your new buffer pool, but not BP32K.

There are other parameters whose values you can change with ALTER TABLESPACE and ALTER INDEX queries. Of these, only the DSETPASS parameters can be changed without damaging the operability of QMF.

Maintaining tables and views using DB2 tables

Anyone with DBA authority can access the DB2 catalog tables to list, display, transfer, or delete tables and views. For complete information on using these DB2 catalog tables, see the appropriate *DB2 UDB SQL Reference* manual.

Transferring ownership of a table or view can be a very difficult task.

Using DB2 catalog tables on z/OS

Note: Certain tables in the system catalog have columns containing binary data. Such columns have character data types but do not contain character data. Retrieving data from these columns can cause an incoherent display, because some of the column “characters” can give unexpected signals to the screen manager.

Listing tables and views

The query in Figure 38 on page 149 returns a list of tables from DB2 UDB for z/OS with columns TABLETYPE (T indicates a table, V indicates a view), TNAME (table name), TABLE SPACENAME, and REMARKS.

```
SELECT TABLETYPE, TNAME, TABLE SPACE NAME, REMARKS
  FROM SYSIBM.SYSTABLES
  WHERE CREATOR = 'userid'
  ORDER BY TABLETYPE, TNAME
```

Figure 38. Listing DB2 tables and views owned by a particular user (OS/390)

Deleting a table or view from the database

Use the SQL DROP TABLE statement or the QMF ERASE command to delete tables or views from the database. Only the creator of the table or someone with DBA authority can delete it.

When you delete the row of the SYSIBM.SYSTABLES table that defines the table, all views, synonyms, and indexes associated with the table are also deleted. Before you drop a table from the database, ensure that no other user relies on it (for example, for command synonym or function key definitions).

For more information on erasing tables, see the appropriate *DB2 UDB Administration Guide*.

Supporting locally defined date/time formats

Note: Locally defined date/time formats are not supported in CICS.

Locally defined date/time formats on z/OS

To define local formats, your installation creates two formatting routines. One of these, named DSNXVDTX, formats dates. The other, named DSNXVTMX, formats times. Creating these routines is a DB2 administration task. If you yourself must do it, see information on locally defined formats in the *DB2 UDB for z/OS Administration Guide*.

Specifying the format

When creating a report, a user can specify the local format for either type of data: TDL for dates; TTL for times. QMF does the formatting by calling the appropriate routine. You must ensure QMF can load both DSNXVTMX and DSNXVDTX.

Making the edit routine available

You can make these routines available by placing their load library in the STEPLIB concatenation of your users' JCL. Make certain that this library is searched before the DB2 program library. If the program library is searched first, QMF loads and uses two IBM-supplied stubs from the DB2 library. These stubs are meant to be used when no local formats are defined: they do no formatting at all. For example, the formatting routines are in the library

XYZ.FORMAT. The library is properly placed in the STEPLIB statement in Figure 39, where the DB2 program library is DSN230.SDSQLOAD.

```
//STEPLIB DD DSN=ISP.V2R2M0.ISPLOAD,DISP=SHR
//        DD DSN=ISR.V2R2M0.ISRLOAD,DISP=SHR
//        DD DSN=QMF710.SDSQLOAD,DISP=SHR
//        DD DSN=XYZ.FORMAT,DISP=SHR          (local formatting library)
//        DD DSN=DSN230.DSNLOAD,DISP=SHR      (DB2 program library)
//        DD DSN=GDDM.OSPID.GDDMLOAD,DISP=SHR
```

Figure 39. Making the edit routine available

Locally defined date/time formats on CICS z/OS

Locally defined date and time edit codes (TTL and TDL) available in other QMF operating environments are not available in QMF on CICS. If you choose to write an edit exit routine to carry out these functions that are not supplied by IBM, you cannot use TTL and TDL as the edit codes. Instead, use Uxxxx or Vxxxx edit codes to identify your local date and time exit routines.

Customizing the document editing interface for users

The document interface is an IBM-supplied macro. Using this macro, a user operating outside QMF can begin a QMF session. In that session, the user can 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.

Customizing the document editing interface on z/OS

The document interface is an IBM-supplied macro for the ISPF/PDF and PS/TSO editors.

Before your users can use this macro, you must:

- Ensure that each user has the proper QMF resources.

  On z/OS, the resources are the QMF libraries. In the sample TSO logon procedure, these have names of the form:

  ```
  QMF810.DSQ*
  ```

  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. This section discusses the changes, both required and optional. To use the document interface, you should also see *Using DB2 QMF*.

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 QMF810.SDSQCLTE on z/OS.
- The other components are members of QMF810.SDSQSAPE on z/OS.

Renaming the document interface macro DSQAED1P

The macro component, DSQAED1P, is the macro that users call to use the document interface.

To use the macro:

- Rename a copy of the macro, preferably to GETQMF. This is the name used for the macro in this publication and in *Using DB2 QMF*.
- Place the renamed copy in QMF810.SDSQCLTE; that is, in the library containing the original.

If you are using an NLF: The main macro is the member DSQA*n*D1P of the library QMF810.DSQCLST*n*. Like the main English-language macro, it 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.

Placing the Q.DSQAED1S procedure in the database

The Q.DSQAED1S procedure is in the member DSQAED1S of the QMF810.SDSQSAPE library. The process of placing the procedure in the database depends on the version of DB2.

As the user Q, you can easily place Q.DSQAEDIS in the database by entering the following QMF command:

```
IMPORT PROC DSQAED1S FROM 'QMF810.SDSQSAPE(DSQAED1S)'  (SHARE=YES
```

If you are not the user Q, but have one of the following:

- SYSADM authority
- SYSCTRL authority
- Q as one of your secondary authorization IDs

you can still easily place DSQAED1S in the database by entering the following QMF commands from the query panel:

```
SET CURRENT SQLID = 'Q'
IMPORT PROC DSQAED1S FROM 'QMF810.SDSQSAPE(DSQAED1S)'  (SHARE=YES
```

A user other than Q who has neither SYSADM (or SYSCTRL) authority nor Q as one of the user's secondary authorization IDs, needs to use the procedure described in "Transferring ownership to Q."

If you are using an NLF: Change the NLID in the member DSQAnD1S of the QMF810.SDSQSAPn library.

Transferring ownership to Q

If you cannot use QMF as the user Q, you can still issue the commands in the previous section. However, you must first transfer ownership of the procedure from your authorization ID to Q. You can do this as follows:

1. Create the following query:

```
UPDATE Q.&T
  SET OWNER = 'Q'
  WHERE NAME = &N AND OWNER = USER
```

2. Run the following commands:

```
RUN  QUERY ( &T=OBJECT_DIRECTORY, &N='DSQAED1S'
RUN  QUERY ( &T=OBJECT_DATA, &N='DSQAED1S'
RUN  QUERY ( &T=OBJECT_REMARKS, &N='DSQAED1S'
```

   Each command updates one of the Q.OBJECT tables and requires the UPDATE privilege on these tables.

If the queries fail to run, an object named Q.DSQAED1S might already be in the database. If so, rename that object or delete it before you attempt to transfer ownership again. One of the following two queries can rename or delete the object for you. You must run the three RUN QUERY commands on whichever query you choose.

- To rename the object, use the following query, replacing *newname* with the new name of the object:

```
UPDATE Q.&T
  SET NAME = 'newname'
  WHERE NAME = &N AND OWNER = 'Q'
```

- To delete the object, use the following query:

```
DELETE FROM Q.&T
   WHERE NAME = &N AND OWNER = 'Q'
```

Changing the data components

There are five data components, all in the library QMF810.SDSQSAPE on z/OS. Unlike the CLISTs and macros, these components contain neither logic nor executable commands. Instead, they contain information that can appear in messages or in the 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 QMF810.SDSQSAPE on z/OS.

If you do, 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 or minidisk.

  If you do, you must copy all the other data components from the old library into the new library on z/OS.

If you use the second method, you must make the change to the macro DSQAED1P or DSQAED2P.

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 NLID in the member DSQA*n*D0L of the QMF810.DSQSAMP*n* library on z/OS.

The DCF components: The DCF (Document Composition Facility) is a licensed IBM text processing system that supports the use of computers in preparing print documentation.

If your installation uses DCF, you might want to change the remaining four DCF components. For more on DCF, see the Document Composition Facility: SCRIPT/VS Text Programmer's Guide.

A user can tell 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 inserted just before the report. In the IBM-supplied component, these are:

```
.* 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 inserted just after each page footing. In the IBM-supplied component, the single furnished statement is:

```
.* QMF Document Interface page footing control:
```

DSQABD03: Supplies statements inserted just before each page heading. In the IBM-supplied component, these are:

```
.PA NOSTART
.* QMF Document Interface page heading control:
```

DSQABD04: Supplies statements inserted just after the end of the report. In the IBM-supplied component, these are:

```
.* QMF Document Interface footing control:
.RE
.* QMF REPORT END
```

Changing the CLISTs, and macros

As mentioned earlier, these components are all in the library QMF810.SDSQCLTE. If you change the CLISTs or macros, change a copy, not the original, and place it in another library. On z/OS, 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 QMF810.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 this:

```
//SYSPROC  DD DSN=SYSUT2.CLIST,DISP=SHR
//         DD DSN=XYZ.NEWCLIST,DISP=SHR
//         DD DSN=QMF810.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 = QMF810.DSQSAMP&LANGCHAR
  SET &BASELIB = QMF810.SDSQSAPE
  ```

  &SAMPLIB
  Identifies the library containing the data components of the document interface

  &BASELIB
  Identifies the QMF sample library

  When &LANGCHAR has the value E, both variables name the same library—QMF810.SDSQSAPE. If the libraries have different names, change the names assigned: &SAMPLIB and &BASELIB.

- Change the 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 this is done, the document interface can reallocate DSQPRINT. This statement restores DSQPRINT to the default. If this is not what you want, 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 necessary to the CLIST code. For example, you might need to add allocations for data sets peculiar to your installation.

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 containing English-language components, as opposed to components for an Uppercase Feature application, for example.

To support LIBDEF allocations, activate LIBDEF service and tailor filenames as necessary:

```
/*****************************************************************@82*/
/* Remove the Following "GOTO NOLIBDEF" statement to allocate     @82*/
/* ISPF libraries using the ISPF LIBDEF service.                  @82*/
/*****************************************************************@82*/
   GOTO NOLIBDEF
/*****************************************************************@82*/
/* ALLOCATE QMF ISPF LIBRARIES USING LIBDEF                       @82*/
/*****************************************************************@82*/
 SET PNAME = 'QMF810.DSQPLIB&LANGCHAR'     /* ISPF Panel Library      */
 SET MNAME = 'QMF810.DSQMLIB&LANGCHAR'     /* ISPF Message Library    */
 SET SNAME = 'QMF810.DSQSLIB&LANGCHAR'     /* ISPF Skeleton Library   */
 SET LNAME = 'QMF810.SDSQLOAD'              /* QMF Modules             */
 ISPEXEC LIBDEF ISPPLIB DATASET ID(&PNAME)
```

Changing DSQABD1P to Support LIBDEF: If you allocated QMF libraries using the LIBDEF function, modify DSQABD1P to free the use of LIBDEF allocated libraries. Uncomment the following statements in DSQABD1P:

```
 /*************************************************************/
 /* FREE ISPF LIBDEFs                                      @82*/
 /* You might or might not need to free libdefs here.         */
 /* If you do, then remove comments from LIBDEF statements.   */
 /*************************************************************/
 /* ISPEXEC LIBDEF ISPPLIB DATASET ID()                       */
 /* ISPEXEC LIBDEF ISPMLIB DATASET ID()                       */
 /* ISPEXEC LIBDEF ISPSLIB DATASET ID()                       */
 /* ISPEXEC LIBDEF ISPLLIB DATASET ID()                       */
 /* FREE FI(DSQLLIB)                                          */
```

Changing DSQABD1C: You can modify this component in the following ways:

- Change the 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 statement's SPACE operand if your users create extremely large reports.

- Change the 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 QMF810. If not, you must add information to the PARM operand of the statement. For example, the subsystem and application plan are named ABC and QMFXXX. Then the modified statement might look like this:

```
ISPEXEC SELECT PGM(DSQQMF&LANGCHAR)
                   PARM(I=&PROCNAME,S=ABC,P=QMFXXX)
                   NEWAPPL(DSQ&LANGCHAR)
```

  The modified statement overrides default values for two of QMF's program parameters.

  For a discussion of program parameters, see Chapter 10, "Customizing your start procedure," on page 79.

Customizing the QMF EDIT command

With the EDIT command, you can modify QMF queries and procedures with an editor. One of these editors can be ISPF/PDF (provided that QMF is started under ISPF).

The EDIT command on z/OS

The following procedure assumes that you use an editor that can be called by a CLIST operating under ISPF. The EDIT TABLE command calls the Table Editor, and does not require a text editor.

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 the CLIST, XYZEDIT, to edit the data set, USERA.XYZDATA.TEXT:

   ```
   XYZEDIT 'USERA.XYZDATA.TEXT'
   ```

2. Place the CLIST in a command library allocated to everyone with access to the editor. Place it in a library that is part of the concatenation for the data set SYSPROC. One possible choice is the QMF library, QMF810.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 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 blocking factor of 51 (BLKSIZE=4029)
4. In the JCL for each user, allocate the data set cataloged for that user in step 3. Allocate it with the ddname `DSQEDIT`. Write `DISP=OLD` for the disposition of the data set.
5. Advise users how to specify the EDIT command. The command has the following format:

   ```
   EDIT yyyy (EDITOR=xxxx)
   ```

   where *yyyy* is either PROC or QUERY, and *xxxx* is the name of the CLIST created to call the editor. For more on the EDIT command, see *DB2 QMF Reference*.
6. You can edit your QMF SQL query or QMF procedure in 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 in the QMF application ID DSQE, or DSQ*n* where *n* is the NLF character. In addition, QMF sets the function keys and location of the command line to fit the QMF product.

   If you need to 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 in an application ID other than DSQ*. To do this, execute two small REXX programs or CLISTs. The first program simply routes execution to the second program, which then invokes the editor running in the desired ISPF application ID with the desired function key or other special setup requirements such as an edit invocation macro or a unique edit panel.

   The REXX program example in Figure 40 on page 158 shows how to edit the SQL query or QMF procedure 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 40. Editing 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, enter the following QMF EDIT command on the QMF command line:

```
EDIT QUERY (E=MYEDIT)
```

Important: If you edit a procedure or query, and the resulting object is too large to fit in QMF's work area, QMF truncates the object and displays an error message. QMF saves the entire object, however, in a file associated with the ddname QMFEDIT. To bring the object into QMF, the user needs to issue a RESET DATA command. This information, including the file name of the saved object, is provided in the message help for the error message associated with this condition.

Enabling English support in an NLF environment

Every NLF has a complete set of translated verbs, keywords, messages, and panels for QMF. The global variable DSQEC__NLFCMD__LANG allows you to change the language in which the user enters commands.

Set DSQEC__NLFCMD__LANG to 1 to allow users to enter commands only in English.

The default value, 0, allows users to enter commands and keywords only in the national language of the current session, except for the following commands:

SET
GET
INTERACT

MESSAGE
START

QMF allows you to enter these commands in either English or the NLF, regardless of how you set DSQEC_NLFCMD_LANG.

Use the DSQEC_FORM_LANG variable to enable users working in an NLF environment to store their form objects in the English language. The LANGUAGE option on the SAVE, EXPORT, and IMPORT commands allows users to specify the national language of the saved form. The values for this option are ENGLISH and SESSION, and are controlled by the global variable DSQEC_FORM_LANG.

Set DSQEC_FORM_LANG to 0 to use the language of the current session as the national language of the saved form.

The default value is 1, which specifies English as the language of the saved form.

If the user specifies the LANGUAGE keyword on the IMPORT or EXPORT command, that value overrides the current value of the DSQEC_FORM_LANG variable.

To change the national language displayed during a QMF session, the QMF user must end the current QMF session and begin another. You cannot change the language from within the QMF session.

Using global variables to define the currency symbol

If you require a currency symbol that is not represented on the keyboard, you can specify the currency symbol by using the HEX value in a Procedure with Logic. For example, the following PROC will set the currency symbol to HEX '9F':

```
/*   */
"SET GLOBAL (DSQDC_CURRENCY =" '9F'X
```

If trailing blanks are needed for the currency symbol, you can put the currency symbol in single quotes as follows:

```
SET GLOBAL (DSQDC_CURRENCY = 'FR '
```

You can use the command in either the command line or in a linear PROC.

Chapter 14. Planning and installing a QMF NLF

A QMF NLF (National Language Feature) is the software that provides you with a QMF environment tailored to a specific language.

In general, QMF functions available in the base English-language session can be performed in a NLF session, and vice-versa.

This chapter parallels the installation steps required for the base QMF product. Where there are significant procedural differences, this chapter explains the procedures for installing the NLF. Where there are job, library, or program name differences, this chapter provides the proper names, but the procedures to follow are explained in the QMF Program Directory.

A module, library, or job name may contain a *n*, representing the National Language Identifier. The *n* symbol is replaced with the actual NLF ID before the product is shipped; you do not need to replace the symbol. (See Table 40 on page 163 for a list of the FMID values for each NLF.)

Profile table and NLF

When you install an NLF, three rows are added to the QMF profile table (Q.PROFILES) to support the NLF. These rows are inserted with a user ID of SYSTEM for the TSO and CICS environments. A unique row is added for each NLF that you install.

The NLF must be installed in each DB2 UDB subsystem you want to use it in. The JCL and control statements for the NLF are shipped on the IBM software distribution (ISD) tape for that feature.

Planning for QMF NLF

This section describes hardware and program product requirements, SMP/E requirements, distribution libraries, target libraries, and user data sets for the NLF.

Hardware and program product requirements

Make sure that your GDDM and ISPF environments, as well as your controllers, terminals, and keyboards, are set up to display the characters for the national language feature you are installing.

SMP/E requirements

Additional DASD space is required for the SMP/E data sets, distribution libraries, target libraries, and user data sets. The DASD space shown here for the distribution, target, and user libraries for a QMF NLF is in addition to what is required for installing the base QMF product. See "Estimating storage" on page 18 for SMP/E requirements for installing base QMF. QMF and its features are added to the SMP/E data sets. For more information on estimating the size of SMP/E and QMF target and distribution library data sets, refer to the QMF Program Directory.

Distribution libraries for QMF NLF

The DB2 QMF Version 8.1 distribution libraries for the NLF are:

- QMF810.ADSQMACn which contains QMF NLF installation procedures, IVP, sample queries, and QMF procedures.
- QMF810.ADSQPMSn which contains ISPF panels for QMF NLF

The QMF NLF distribution libraries and the additional estimated DASD space required (in cylinders) is shown in Table 38.:

Table 38. Additional DASD space for QMF NLF distribution libraries (cylinders)

DSNAME	Content	3380	3390	9345
QMF810.ADSQMACn	QMF NLF install procs	15	13	15
QMF810.ADSQPMSn	QMF NLF ISPF panels	1	1	1

Target libraries for QMF NLF

Estimated additional DASD space required (in cylinders) for the QMF NLF target libraries is shown in the table below:

Table 39. Additional DASD space for QMF NLF target libraries (cylinders)

DSNAME	3380	3390	9345
QMF810.SDSQSAPn	17	15	17
QMF810.SDSQPLBn	1	1	1
QMF810.SDSQCLTn	2	1	2
QMF810.SDSQMLBn	1	1	1
QMF810.SDSQEXCn	1	1	1
QMF810.SDSQUSRn			

IBM software distribution (ISD) tape

To install a QMF NLF, first read in information from the IBM ISD tape. The tape contains:

- SMP/E control statements
- JCLIN for QMF 8 NLF
- JCL for installation-verification procedures
- Programs in load module format
- Panels and other items used by DB2 QMF Version 8.1 NLF

The ISD tape has an SMP/E (RELFILE) format. The format is described in the *OS/390 System Modification Program Extended Reference*.

FMID

A function modification identifier (FMID) identifies QMF NLF to SMP/E. The language identifier and FMID for each NLF are provided in Table 40.

Table 40. Language ID and FMID

National Language Feature	Language ID	QMF 8.1 FMID
U/C English	U	JSQ8851
Danish	Q	JSQ8855
French	F	JSQ8856
German	D	JSQ8857
Italian	I	JSQ8858
Japanese Kanji	K	JSQ8859
Korean Hangeul	H	JSQ885A
Brazil Portuguese	P	JSQ885B
Spanish	S	JSQ885C
Swedish	V	JSQ885D
Swiss French	Y	JSQ885E
Swiss German	Z	JSQ885F
Canadian French	C	JSQ885G

SMP/E associates all modifications of a program to a system release level (SREL) of that program. The system release level for QMF is P115.

All the files on the tape, except the first, are IEBCOPY unloaded partitioned data sets and correspond to the NLF distribution libraries. The first data set contains SMP/E control statements for NLF. This tape contains all the procedures and data required for installation.

The installation process

The installation steps are outlined on the following pages.

The NLF JCL and control statements are shipped on the ISD tape. You must complete an SMP/E installation for each QMF NLF to be installed before performing the steps that are listed in this chapter. The SMP/E installation instructions are contained in the QMF Program Directory.

The NLF requires the use of the DB2 QMF Version 8.1 sample library, QMF810.SDSQSAPE, and the load module library, QMF810.SDSQLOAD.

Preliminary: read the program directory and complete the NLF worksheet

Before beginning the installation process, read the NLF program directory for supplementary data. Because the program directory is updated between releases of QMF NLF, it may contain useful information, including descriptions of PTFs and APARs, as well as modifications to this book that may have occurred since its publication date. The table below shows the information that you will have to provide during a QMF NLF installation. You can use this as a worksheet.

Table 41. QMF NLF installation parameters (Version 8.1)

Parameter	Value
Target Library Prefix (Default = QMF810)	
Distribution Library Prefix (Default = QMF810)	
Target Library Volume (Default = xxxxxx)	
Distribution Library Volume (Default = yyyyyy)	
SMP/E Data Set Prefix (Default = IMSVS)	
Local DB2 Subsystem ID (Default=DSN)	
Local DB2 Release Level (Default=V8R1)	
Local DB2 Exit Library (Default=DSN810.SDSNEXIT)	
Local DB2 Load Library (Default=DSN810.SDSNLOAD)	
QMF application plan ID (Default=QMF810)	
QMF table space catalog alias (Default=QMFDSN)	
QMF table spaces volume	
DB2 default punctuation	, (comma) or . (period)
Previous QMF NLF level (migration installs only)	3.3, 6.1, 7.1, 7.2 or NONE

Table 42. QMF NLF Installation Parameters (Version 8 Worksheet-Part 2)

Parameter	Primary	Secondary
Gather the following information, if the scope of the database install is not "R".		
QMF Control Table table space Sizes: (in 1K units) Table Space name Default size (primary, secondary) - Q.COMMAND_SYNONYMS_n (100,20) (see note)	_____	_____
Sizes: (in 1K units) Table Index name Default size (primary, secondary) - Q.COMMAND_SYNONYMSX_n (100,20) (see note)	_____	_____

Before performing the following steps, you must first install QMF NLF in your z/OS environment using SMP/E as documented in the QMF Program Directory.

Installing a QMF NLF

The DSQ1nINS installation CLIST and its panels in addition to jobs DSQ1nMAP, DSQ1nVE, and DSQ1nJVC are not available in DB2 QMF Version 8.1; a QMF NLF can only be installed by submitting batch installation jobs. The installation VSAM panel job DSQ1nPNL and its information are now found in Section 6.2 in the QMF Program Directory.

Note: The QMF NLF installation described in this chapter will satisfy either a QMF New Function mode or a QMF Compatibility mode installation.

Step 1A Update QMF control tables

- If no previous QMF NLF release is installed, do "Substep 1Aa—without a previous QMF NLF release" on page 166
- If QMF NLF 3.3 or higher is in this DB2 subsystem, skip to "Step 1B—Delete earlier QMF NLF sample tables" on page 167

For all these steps that run TSO batch, check the step completion code in the system messages. Completion messages can be found in the SYSTSPRT or the SYSTERM output, as indicated. SYSPRINT provides additional diagnostic information for IBM support.

Substep 1Aa—without a previous QMF NLF release

Perform this step if you do not have a previous QMF NLF installed.

On this step, perform these tasks:

- Add NLF entries in the Q.PROFILES table. The job is QMF810.SDSQSAPn(DSQ1nUPO).
- Create a command synonyms table named Q.COMMAND_SYNONYM_n for the NLF environment. The job is QMF810.SDSQSAPn(DSQ1nCCS).

Preparation: Change the job statements for DSQ1nUPO and DSQ1nCCS to fit your installation. The value of the USER parameter in the job statement is currently "Q" for the owner of the QMF tables. Change this value to your primary authorization ID if your authorization ID is not Q.

Make the necessary changes to the following parameter values in the job's instream procedure:

Parameter name
Description of value (default in parenthesis)

QMFTPRE
Prefix of the QMF target libraries (**QMF810**)
DB2EXIT
Name of the DB2 exit library (**DSN810.SDSNEXIT**)
DB2LOAD
Name of the DB2 program library (**DSN810.SDSNLOAD**)
RGN Job step region size (**2048K**)

DB2 authority: If you are the user Q, run the following query to give you enough authority to run the jobs:

```
GRANT CREATETAB ON DATABASE DSQDBCTL TO Q
```

You may need the query if the database DSQDBCTL was not created by the user Q.

If you are not the user Q, run the following queries, to give you enough authority to run the jobs:

```
GRANT INSERT, UPDATE ON TABLE Q.PROFILES TO authid
GRANT CREATETAB ON DATABASE DSQDBCTL TO authid
```

where *authid* is your primary authorization ID.

Execution: Run the appropriate jobs:
- DSQ1nUPO, to add a line to Q.PROFILES
- DSQ1nCCS, to run required SQL statements

Review SYSTERM for completion messages. If errors occur, examine SYSTSPRT and SYSPRINT for error messages.

Rerunning the job: If the job fails, you can correct the error and rerun it.

Step 1B and 1C—Establish the QMF NLF sample tables

Skip steps 3B and 3C if either of the following conditions apply:

- The NLF is the upper case feature (UCF).
- You already have the sample tables installed from an earlier release of QMF NLF.

These two steps establish the QMF NLF sample tables. The first step drops previously created tables, the second step installs new ones. In the event of a failure, you can restart both of these steps because database changes are not committed until the job that was run by the step ends.

Step 1B—Delete earlier QMF NLF sample tables

Run this step if you are installing QMF 8 NLF into a DB2 subsystem that also contains a previous release of QMF NLF. Otherwise, skip to "Step 1C—Create the NLF sample tables" on page 168

This step deletes the sample tables that were created when the earlier version was installed. The QMF NLF sample tables have been modified for QMF 8 NLF.

Preparation: The job used in this step is QMF810.SDSQSAPn(DSQ1nDSJ). If the tailoring performed was not sufficient, change the job statement to conform to your installation's requirements. If necessary, change the installation parameter values in the job's instream procedure:

Parameter name
Description of value (Default in parenthesis)

QMFTPRE
The prefix name of the QMF target libraries (**QMF810**)
DB2EXIT
Name of the DB2 exit library (**DSN810.SDSNEXIT**)
DB2LOAD
Name of the DB2 program library (**DSN810.SDSNLOAD**)
RGN Job-step region size (**2048K**)

Make no other modifications to the job.

DB2 authorization: If you are not the user Q, run the following query to grant you the necessary authority:

```
GRANT SYSADM TO authid
```

where *authid* is your primary authorization ID.

Execution: Run job DSQ1nDSJ (in the library QMF810.SDSQSAPn). Review SYSTERM for completion messages. If errors occur, examine SYSTSPRT and SYSPRINT for error messages.

Rerunning the job: If the job fails, you can correct the error and rerun it. However, the job may fail because the tables it is trying to drop have already been dropped.

Step 1C—Create the NLF sample tables

This step creates the NLF sample tables.

Note: QMF NLF users at locations within the network are authorized to use all the sample tables created at the location on which you are installing the QMF NLF.

Preparation: The job for this step is QMF810.SDSQSAPn(DSQ1nIVS). If necessary, change the values for the installation parameters in the job's instream procedure:

Parameter name
Description of value (Default in parenthesis)

QMFTPRE
The prefix for the QMF target libraries (**QMF810**)

DB2EXIT
Name of the DB2 exit library (**DSN810.SDSNEXIT**)

DB2LOAD
Name of the DB2 program library (**DSN810.SDSNLOAD**)

RGN Job-step region size (**2048K**)

CDS, CDP
Identify the punctuation mark for the decimal point used in decimal fractions. This must match the DECPOINT option that was specified when DB2 UDB was installed:

- For a period, leave the current values as they are.
- For a comma, change CDS to **6** and CDP to **7**.

For more information on the DECPOINT option, see the *DB2 UDB for z/OS Installation Guide.*

DB2 authority: If you are the user Q, you will need DB2 authority granted by the following SQL statements:

```
GRANT SELECT ON SYSIBM.SYSTABLES TO Q WITH GRANT OPTION
GRANT SELECT ON SYSIBM.SYSTABAUTH TO Q WITH GRANT OPTION
GRANT SELECT ON SYSIBM.SYSCOLUMNS TO Q WITH GRANT OPTION
```

If you are not the user Q, run the following query to give you the necessary authority:

```
GRANT SYSADM TO authid
```

where *authid* is your primary authorization ID.

Execution: Run job DSQ1nIVS in the library QMF810.SDSQSAPn. Review SYSTERM for completion messages. If errors occur, examine SYSTSPRT and SYSPRINT for error messages.

Rerunning the job: If the job fails, correct the error and rerun the job.

If you are installing a QMF NLF into another database, go to "Step 6—Set Up NLF batch job to run batch IVP (optional)" on page 175.

You are now ready to tailor NLF QMF for TSO or CICS.

- For information on tailoring QMF NLF for TSO, see the next section.
- For information on tailoring QMF NLF for CICS, see "Step 3—Tailor NLF QMF for CICS" on page 170.

Step 2—Tailor NLF QMF for TSO

To create a TSO logon procedure for NLF, first make a copy of the TSO logon procedure for the QMF base product.

Except for the following changes to the TSO logon procedure, the procedure for tailoring NLF QMF for TSO is outlined in Chapter 4, "Tailoring QMF for TSO," on page 27.

- The following NLF libraries should be concatenated in front of the QMF base libraries.
  - The statement to concatenate to the ADMGGMAP DD statement is:
    ```
    //ADMGGMAP  DD DSN=QMF810.DSQMAPn,DISP=SHR
    ```
  - The statement to concatenate to the ISPPLIB DD statement is:
    ```
    //ISPPLIB  DD DSN=QMF810.SDSQPLBn,DISP=SHR
    ```
  - The statement to concatenate to the ISPMLIB DD statement is:
    ```
    //ISPMLIB  DD DSN=QMF810.SDSQMLBn,DISP=SHR
    ```
  - The statement to concatenate to the SYSPROC DD statement is:
    ```
    //SYSPROC  DD DSN=QMF810.SDSQCLTn,DISP=SHR
    ```
  - The statement to concatenate to the SYSEXEC DD statement is:
    ```
    //SYSEXEC  DD DSN=QMF810.SDSQEXCn,DISP=SHR
    ```
  - The statement to concatenate to the DSQPNLn DD statement is:
    ```
    //DSQPNLn  DD DSN=QMF810.DSQPNLn,DISP=SHR
    ```
- The statement to start QMF with ISPF looks like the following example:
  ```
  ISPSTART PGM(DSQQMFn) NEWAPPL(DSQn) PARM(DSQSSUBS=dbname,...)
  ```

The ISPF Master Application Menu should be changed as shown in the following figure (DSQQMFn is the NLF program).

```
%------------------------  MASTER APPLICATION MENU  ---------------------
%SELECT APPLICATION ===>_;OPT     +
%                                                          +USERID   -
%                                                          +TIME     -
%   1 +SPF        - SPF PROGRAM DEVELOPMENT FACILITY       +TERMINAL -
+   2 +QMF        - QMF QUERY MANAGEMENT FACILITY          +PF KEYS  -
%   3 +QMFn       - QMF NATIONAL LANGUAGE FEATURE
%
%
%
%
%
%
%   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(DSQQMFE) NEWAPPL(DSQE)'
                3,'PGM(DSQQMFn) NEWAPPL(DSQn)'
           /*                              */
           /* ADD OTHER APPLICATIONS HERE */
           /*                              */
                P,'PANEL(ISPOPT)'
                X,'EXIT'
              ' ',' '
                *,'?' )
)END
```

Figure 41. QMF dialog on ISPF Master Application Menu for NLF

- The statement to start QMF without ISPF looks like the following:

  ```
  DSQQMFn DSQSPLAN=planid,DSQSSUBS=dbname,...
  ```

  where DSQQMFn is the NLF program.

Step 3—Tailor NLF QMF for CICS

You can run this step after the QMF product has been tailored for CICS as described in Chapter 5, "Tailoring QMF for CICS," on page 35. Run all the steps if you are migrating from Version 3.3.

Step 3A—Add NLF QMF transaction ID to DB2 RCT

The database plan ID and authorization ID for a transaction are specified in the DB2 resource control table (RCT). For example, to specify a transaction ID of "QMFn" and an authorization ID of "DEPT1", add the following statement:

```
DSNCRCT TYPE=ENTRY,TXID=QMFn,PLAN=QM720,AUTH=DEPT1
```

QMF ships a sample RCT entry located in QMF810.SDSQSAPn(DSQ1nRCT).

After the RCT is updated with information describing the QMF transaction to DB2, you must then regenerate your RCT.

Step 3B—load QMF GDDM map sets to the ADMF data set

Preparation: The job used in this step is QMF810.SDSQSAPn(DSQ1nADM). Change the job statement to conform to your installation. If necessary, change the values for the installation parameters in the job's instream procedure.

Table 43. Installation parameters for DSQ1nADM

Parameter name	Description of value	Default
QMFTPRE	The prefix name of the QMF target libraries	QMF810
REG	The job-step region size	2048
GDDM	The name of the GDDM ADMF data set	GDDM.ADMF

Step 3C—Update CICS control tables (CICS ESA only)

Before you can run the NLF/QMF feature under CICS, QMF entries must be defined in the CICS system definition file (CSD).

Preparation: The job used in this step is QMF810.SDSQSAPn(DSQ1nCSD). Change the job statement to conform to your installation. If necessary, change the values for the installation parameters in the job's instream procedure:

Table 44. Installation parameters for DSQ1nCSD

Parameter name	Description of value	Default
QMFTPRE	The prefix name of the QMF target libraries	QMF810
REG	The job-step region size	2048
OUTC	The job output class	*
CLOAD	The name of the CICS load library	CICS.SDFHLOAD
CCSD	The name of the CICS CSD data set	CICS.DFHCSD

Step 3D—Update CICS region job stream

The QMF panel file must be added to the existing JCL that is used to start the CICS region containing QMF. Add the following statement:

```
//DSQPNLn DD DSN=QMF810.DSQPNLn,DISP=SHR
```

where *n* is the NLF character.

Step 3E—Run the IVP

Run the IVP as indicated in "Run the IVP (CICS)" on page 53, changing the following names:

- QMF320.DSQSAMPE to QMF810.SDSQSAPn
- DSQ1EIVC to DSQ1nIVC

where *n* is the NLF character.

Step 4—Tailoring QMF NLF for a Workstation Database Server (optional)

QMF support for Workstation Database Server is optional. Perform the steps described in this step only if you intend to run a Workstation Database Server as an application server for your QMF NLF.

Before you install a QMF NLF into a Workstation Database Server from z/OS, you need to verify that you have followed the steps to install the QMF base product into your Workstation Database Server database. The installation of a QMF NLF requires that the outbound Workstation Database Server ID has SYSADM authority. For more information about installing QMF into a Workstation Database Server, see Chapter 6, "Configuring remote servers for QMF Compatibility mode," on page 41.

Check the step completion codes in the system messages. Completion messages can be found in the SYSTSPRT or the SYSTERM output, as indicated. SYSPRINT provides additional diagnostic information for IBM support.

Step 4A—Create QMF NLF control tables in a Workstation Database Server

This step creates QMF NLF command synonym tables and profile rows in a Workstation Database Server.

1. Edit QMF810.SDSQSAPE(DSQ1nDJ2).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1TBJ4 PROC RGN='2048K',              Job-step region size
//             QMFTPRE='QMF810',          Prefix for QMF target libraries
//             DB2EXIT='DSN810.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN810.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for z/OS subsystem ID.
4. Submit job QMF810.SDSQSAPE(DSQ1nDJ2).
5. Check for a return code of 0 or 4. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero or four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and then re-run this job.

Step 4B—Create QMF NLF sample tables in a Workstation Database Server

This step creates the QMF NLF sample tables in a Workstation Database Server.

1. Edit QMF810.SDSQSAPE(DSQ1nDJ4).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1TBJ4 PROC RGN='2048K',              Job-step region size
//             QMFTPRE='QMF810',          Prefix for QMF target libraries
//             DB2EXIT='DSN810.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN810.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for z/OS subsystem ID.
4. Submit job QMF810.SDSQSAPE(DSQ1nDJ4).
5. Check for a return code of 0 or 4. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero or four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Deleting QMF NLF from a Workstation Database Server

This section describes how to delete QMF NLF from a Workstation Database Server.

Deleting QMF from a Workstation Database Server: This step should be run only if you are re-installing QMF into a Workstation Database Server that already contains QMF.

Attention: This step will delete the QMF NLF command synonym tables and system profile rows from a Workstation Database Server.

1. Edit QMF810.SDSQSAPE(DSQ1nDX1).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1TBJ4 PROC RGN='2048K',              Job-step region size
//             QMFTPRE='QMF810',          Prefix for QMF target libraries
//             DB2EXIT='DSN810.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN810.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for z/OS subsystem ID.
4. Submit job QMF810.SDSQSAPE(DSQ1nDX1).
5. Check for a return code of 0 or 4. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero or four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Deleting QMF NLF sample tables from a Workstation Database Server: This step should be run only if you are reinstalling the QMF NLF into a Workstation Database Server that already contains the QMF NLF.

This step will drop and create all QMF NLF sample tables and table space from a Workstation Database Server.

1. Edit QMF810.SDSQSAPE(DSQ1nDX2).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1TBJ4 PROC RGN='2048K',              Job-step region size
//             QMFTPRE='QMF810',          Prefix for QMF target libraries
//             DB2EXIT='DSN810.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN810.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for z/OS subsystem ID.
4. Submit job QMF810.SDSQSAPE(DSQ1nDX2).
5. Check for a return code of 0 or 4. Review SYSTERM for completion messages.

   Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and then re-run this job.

Step 5—Tailoring QMF NLF for a DB2 UDB for iSeries server (optional)

QMF support for DB2 UDB for iSeries Database Servers is optional. Run the steps described here only if you intend to run a DB2 UDB for iSeries Database Server as an application server for your QMF NLF. Before you install a QMF NLF into a DB2 UDB for iSeries Database Server from z/OS, you must verify that you have followed the steps needed to install the QMF base product into your DB2 UDB for iSeries Database Server database.

Check the step completion codes in the system messages. Completion messages can be found in the SYSTSPRT or the SYSTERM output, as indicated. SYSPRINT provides additional diagnostic information for IBM support.

Create QMF NLF control table updates in a DB2 UDB for iSeries server.

1. Edit QMF810.SDSQSAPE(DSQ1nAS2).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1nAS2 PROC RGN='2048K', Job-step region size
// QMFTPRE='QMF810', Prefix for QMF target libraries
// DB2EXIT='DSN810.SDSNEXIT', Exit DB2 library name
// DB2LOAD='DSN810.SDSNLOAD' DB2 program library name
```

3. Change in SYSTEM() to your DB2 UDB for z/OS subsystem ID.
4. Carefully read the comments in the job and make any necessary changes.
5. Submit job QMF810.SDSQSAPE(DSQ1nAS2).
6. Check for a return code of 0 or 4. Review SYSTERM for completion messages. Do not proceed if the return code is other than zero or four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Create QMF NLF sample tables in a DB2 UDB for iSeries server

1. Edit QMF810.SDSQSAPE(DSQ1nAS4).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1nAS4 PROC RGN='2048K', Job-step region size
// QMFTPRE='QMF810', Prefix for QMF target libraries
// DB2EXIT='DSN810.SDSNEXIT', Exit DB2 library name
// DB2LOAD='DSN810.SDSNLOAD' DB2 program library name
```

3. Change in SYSTEM() to your DB2 UDB for z/OS subsystem ID.
4. Carefully read the comments in the job and make any necessary changes.
5. Submit job QMF810.SDSQSAPE(DSQ1nAS4).
6. Check for a return code of 0 or 4. Review SYSTERM for completion messages. Do not proceed if the return code is other than zero or four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Step 6—Set Up NLF batch job to run batch IVP (optional)

For the NLF, you must modify the TSO logon procedure described in "Set up a QMF batch job to run batch IVP (optional)" on page 34. Modify the ISPSTART command at the end of that procedure:

```
ISPSTART PGM(DSQQMFn) NEWAPPL(DSQn) PARM(DSQSMODE=B,DSQSRUN=Q.DSQ1nBAT)
```

Step 7—Running the IVP for QMF interactive mode

See "Run the IVP (TSO)" on page 51 and "Run the IVP (CICS)" on page 53 for information on running the IVP. The NLF IVP (DSQ1nIVP) found in the library QMF810.SDSQSAPn is used to verify the NLF. This procedure (DSQ1nIVP) imports a query from the QMF English sample library (*prefix*.SDSQSAPE), where *prefix* is the prefix for the QMF data sets.

The procedures were written assuming that this prefix is QMF810. If this is not your prefix, change QMF810 to match your prefix wherever it appears in the DSQ1nIVP procedure.

```
IMPORT PROC FROM 'QMF810.SDSQSAPn(DSQ1nIVP)'
RUN PROC
```

Step 8—Installing the national language sample queries and procedures

After the QMF NLF is installed and verified, use it to install the translated versions of the sample queries and procedures. Do this in two steps:

- "Step 8A—Deleting the existing sample queries and procedures"
- "Step 8B—Installing the national language sample queries and procedures" on page 177

Step 8A—Deleting the existing sample queries and procedures

Skip this step if you do not have a previous release of the QMF NLF with the same language identifier installed at your location.

To delete the existing sample queries and procedures, import and run the QMF procedure DSQ1nSQD (from the QMF Version 8.1 sample library, QMF810.SDSQSAPn), using translated QMF commands where appropriate. This procedure (DSQ1nSQD) imports a query from the QMF English sample library (*prefix*.SDSQSAPE), where *prefix* is the prefix for the QMF data sets.

The procedures were written assuming that this prefix is QMF810. If this is not your prefix, change QMF810 to match your prefix wherever it appears in the DSQ1nSQD procedure.

```
IMPORT PROC FROM 'QMF810.SDSQSAPn(DSQ1nSQD)'
RUN PROC
```

You may see the Database Status panel when you perform this step. You are not required to perform any action because of it.

DB2 authorization: If you are the user Q, you already have the necessary authority.

If you are not the user Q, run the following query to give you the necessary authority:

```
GRANT UPDATE ON Q.OBJECT_DIRECTORY TO authid
GRANT UPDATE ON Q.OBJECT_REMARKS TO authid
GRANT UPDATE ON Q.OBJECT_DATA TO authid
```

where *authid* is your primary authorization ID.

Restarting this step: If the job fails, proceed to the next step.

Step 8B—Installing the national language sample queries and procedures

To install the national language sample queries and procedures, import and run the QMF procedure in QMF810.SDSQSAPn (DSQ1nSQI), using translated QMF commands where appropriate. This procedure (DSQ1nSQI) imports a query from the QMF English sample library (*prefix*.SDSQSAPE), where *prefix* is the prefix for the QMF data sets.

The procedures were written assuming that this prefix is QMF810. If this is not your prefix, change QMF810 to match your prefix wherever it appears in the DSQ1nSQI procedure.

```
IMPORT PROC FROM 'QMF810.SDSQSAPn(DSQ1nSQI)'
RUN PROC
```

If you are not the user Q, see Install the QMF application queries and procedures (TSO) for the necessary GRANT queries you must run.

This step also installs the batch mode IVP and sample application procedures.

DB2 authorization: If you are the user Q, you already have the necessary authority.

If you are not the user Q, run the following query to give you the necessary authority:

```
GRANT UPDATE ON Q.OBJECT_DIRECTORY TO authid
GRANT UPDATE ON Q.OBJECT_REMARKS TO authid
GRANT UPDATE ON Q.OBJECT_DATA TO authid
```

where *authid* is your primary authorization ID.

Restarting this step: If a failure occurs during this job, correct the error and run procedure DSQ1nSQD which deletes any previously created sample queries. Then rerun procedure DSQ1nSQI.

Step 9—Running the batch-mode IVP (optional)

See “Optional: Run the batch-mode IVP” on page 57 for information on running the batch IVP. Start the batch IVP by using the national language program, DSQQMFn, instead of DSQQMFE. This step uses the QMF batch IVP.

Step 10—Post-installation cleanup

See "Clean up after installation" on page 58 for information on cleanup activities following installation.

Skip this step if you do not already have an earlier release of the QMF NLF installed.

You may want to delete libraries of an earlier QMF NLF release. These are listed in the following figure with their default prefixes.

Attention: Pay special attention to the prefix to avoid deleting a QMF Version 8.1 data set.

```
V2R2 Data Sets   V2R3 Data Sets    V2R4 Data Sets    V3R1 Data Sets

 QMF220.DSQMACn    QMF230.DSQMACn    QMF240.DSQMACn    QMF310.DSQMACn
 QMF220.DSQPMSn    QMF230.DSQPMSn    QMF240.DSQPMSn    QMF310.DSQPMSn
 QMF220.DSQSAMPn   QMF230.DSQSAMPn   QMF240.DSQSAMPn   QMF310.DSQSAMPn
 QMF220.DSQMAPn    QMF230.DSQMAPn    QMF240.DSQMAPn    QMF310.DSQMAPn
 QMF220.DSQCLSTn   QMF230.DSQCLSTn   QMF240.DSQCLSTn   QMF310.DSQCLSTn
 QMF220.DSQPLIBn   QMF230.DSQPLIBn   QMF240.DSQEXECn   QMF310.DSQEXECn
 QMF220.DSQSLIBn   QMF230.DSQSLIBn   QMF240.DSQUSERn   QMF310.DSQUSERn
 QMF220.DSQMLIBn   QMF230.DSQMLIBn   QMF240.DSQPLIBn   QMF310.DSQPLIBn
 QMF220.DSQTLIBn   QMF230.DSQTLIBn   QMF240.DSQSLIBn   QMF310.DSQSLIBn
                                     QMF240.DSQMLIBn   QMF310.DSQMLIBn
                                     QMF240.DSQTLIBn   QMF310.DSQTLIBn


 V3R1M1 Data Sets  V3R2 Data Sets    V3R3 Data Sets    V6R1 Data Sets

 QMF311.DSQMACn    QMF320.DSQMACn    QMF330.DSQMACn    QMF610.ADSQMACn
 QMF311.DSQPMSn    QMF320.DSQPMSn    QMF330.DSQPMSn    QMF610.ADSQPMSn
 QMF311.DSQSAMPn   QMF320.DSQSAMPn   QMF330.DSQSAMPn   QMF610.SDSQSAPn
 QMF311.DSQMAPn    QMF320.DSQMAPn    QMF330.DSQMAPn    QMF610.SDSQPLBn
 QMF311.DSQCLSTn   QMF320.DSQCLSTn   QMF330.DSQCLSTn   QMF610.SDSQCLTn
 QMF311.DSQEXECn   QMF320.DSQEXECn   QMF330.DSQEXECn   QMF610.SDSQMLBn
 QMF311.DSQUSERn   QMF320.DSQUSERn   QMF330.DSQUSERn   QMF610.SDSQEXCn
 QMF311.DSQPLIBn   QMF320.DSQPLIBn   QMF330.DSQPLIBn   QMF610.SDSQUSRn
 QMF311.DSQSLIBn   QMF320.DSQSLIBn   QMF330.DSQSLIBn   QMF610.DSQMApn
 QMF311.DSQMLIBn   QMF320.DSQMLIBn   QMF330.DSQMLIBn
 QMF311.DSQTLIBn   QMF320.DSQTLIBn   QMF330.DSQTLIBn
```

Figure 42. Libraries to be deleted from earlier QMF NLF releases

Step 11—Accept the permanent libraries

Perform this step if this is a first-time QMF NLF install for language *n* in a z/OS system.

The job name for this step is DSQ1nJAC, which invokes procedure DSQ1nJSM or the SMP/E procedure used at your installation.

Step 12—Create a cross-CDS environment

Skip this step if no maintenance changes were made to modules common to base QMF Version 8.1 and the NLF. This step allows SMP/E to keep track of changed modules.

This step contains an SMP/E job to update the JCLIN data in the SMP/E environment. This job is located in member DSQ1nCDS (in library QMF810.SDSQSAPn). The input to this job is located in member DSQ1nJCL (in library QMF810.SDSQSAPn).

Chapter 15. Enabling users to print objects

QMF end users frequently need to print data they retrieve from the database. This data might be in the format of a report, a chart, a 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 QMF objects you need to print. This chapter 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.

If you need to print double-byte character set (DBCS) data, you can use the DSQSDBCS program parameter when you start QMF to allow users to print DBCS data from non-DBCS terminals.

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 section 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 physical 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.

CICS considerations

These considerations are for CICS:

- 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 CICS-supplied transaction CEBR.
- In CICS, if you need to print more than 32,767 rows of output, use GDDM or define transient data queues to use with QMF.

  Temporary storage queues cannot handle more than 32,767 rows of data.

Using GDDM services to handle printing

Important: The explanations in this section apply only if you are using the GDDM default values shipped with the GDDM product. For more information on changing these values, see *GDDM System Customization and Administration*.

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 FSFRCE GDDM services. When printing is complete, QMF closes the print operation with a DSDROP statement.

GDDM services on z/OS

These services apply to native z/OS batch, TSO, ISPF, and CICS.

Native z/OS batch and TSO

To use GDDM services for printing QMF objects, you must:

1. Choose a GDDM nickname for the print device, as explained in "Choosing a GDDM nickname for your printer" on page 183.

   Nicknames enable you to predefine 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.

2. Update the GDDM defaults module, ADMADFT, with the specifications of your nickname.
3. Allocate the ddname ADMDEFS. Allocating the ddname ADMDEFS is explained in "Allocating the nickname file for native z/OS batch, TSO, and ISPF" on page 189.
4. Update the PRINTER field of the user's row in the Q.PROFILES table.

CICS

To use GDDM services for printing QMF objects, you must:

1. Choose a GDDM nickname for the print device.

   Nicknames enable you to predefine 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.
2. Update the GDDM defaults module, ADMADFC, with the specifications of your nickname.
3. Update CICS resource definitions with the values in the nickname specification, so that CICS can link the nickname with the physical device it manages.
4. Update the PRINTER field of the user's row in the Q.PROFILES table.

Choosing a GDDM nickname for your printer

Here is information on the data sets GDDM searches for.

Native z/OS batch, TSO, and ISPF: In native z/OS batch and TSO, 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, ADMADTC, for a matching nickname that defines how and where to direct the output.

CICS: In CICS, GDDM searches only the defaults module, ADMADTC. GDDM uses nicknames to recognize all the devices with which it can communicate (including terminals).

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 terminals.
- **Family 2 devices** include devices such as IBM 3270 terminals and queued printers.

- **Family 3 devices** are system printers that support the ANSI code of carriage control characters.
- **Family 4 devices** are advanced function printers for which you need to use the ADMOPUT and ADMOPUJ utilities (in TSO and native z/OS batch only) to print output. These utilities are provided by GDDM.

This chapter explains how to define nicknames for Family 1, 2, and 3 devices. For more information on how to set up a nickname for a Family 4 printer and use the ADMOPUT and ADMOPUJ utilities, see *GDDM System Customization and Administration*.

Creating the nickname specification

Here are the instructions to create nicknames on native z/OS, TSO, and CICS.

Native z/OS batch, TSO, and ISPF: Add the nickname to your ddname ADMDEFS data set. GDDM looks at this data set first. If the nickname is not found, GDDM looks in the external default module, ADMADFT, in which you define a GDDM ADMMNICK specification.

CICS: To create a nickname in CICS, first define a GDDM ADMMNICK specification in the GDDM external default 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.

Use the format shown in Figure 43 for your ADMMNICK specification.

```
ADMMNICK NAME=nickname,TOFAM=family_type,DEVTOK=device_token(,TONAME=name)
```

Figure 43. Using the ADMMNICK specification to define a nickname

TONAME is used only in CICS.

- 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 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**?** 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 (for example, a 3820 printer that prints 60 rows by 85 columns, 6 lines per inch). For a list of valid device tokens, see *GDDM System Customization and Administration*.

- In CICS, the TONAME field points to entries in the TCT or DCT so that CICS is able to properly manage communication between GDDM and the printer. Use TONAME to point to the name of a 1-character to 4-character printer definition name with a value that depends on the type of device:
  - If the nickname defines a Family 1 or 2 printer, TONAME points to a matching entry in the CICS terminal control table (TCT), which defines the printer to CICS. In the matching entry, the TRMIDNT field has the same value as TONAME.

    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 the nickname defines a Family 3 printer, TONAME points to a matching entry in the CICS destination control table (DCT), which defines the printer to CICS. In the matching entry, the DESTID field has the same value as TONAME.

A unique label can be added to the syntax. For example, GDDMPRT1 is a possible label for the nickname definition:

```
GDDMPRT1 ADMMNICK NAME=MYPRINT,TOFAM=3,DEVTOK=ADMKSYSP
```

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 Figure 44. 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
```

Figure 44. Using the ADMMNICK specification to define a nickname for a Family 2 printer

Native z/OS batch, TSO and ISPF: After you create your nickname in TSO 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 already exists. This data set is `userid.ADMPRINT.REQUEST.#nnnnn`, where `nnnnn` is a sequence number. You can then print the data set 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 using GDDM nicknames, the QMF-supplied GDDM map groups must be made available to the GDDM print utility. The ADMGGMAP DD statement contains the name of the data set (QMF810.SDSQMAPE) that holds the map groups:

```
//ADMGGMAP DD DSN=QMF810.SDSQMAPE,DISP=SHR
```

Without this statement, any attempt to print a form on a Family 2 printer ends in an error. For more information on the GDDM print utilities, see *GDDM System Customization and Administration*.

Important: In CICS, after you create the ADMMNICK specification, link the name with a physical device by updating the TCT. Make sure TONAME in the ADMMNICK specification and TRMIDNT in the TCT have matching values.

You can also use CICS RDO facilities to update the CSD online. If you define the printer this way, make sure the TERMINAL attribute in the CSD and TONAME in the ADMMNICK specification have matching values.

Example nickname for a family 3 GDDM printer

Use this information to define the nickname for a family 3 GDDM printer on native z/OS batch and TSO.

Native z/OS batch, TSO and ISPF: To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one in below.

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S,TONAME=370P (CICS)
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S (CMS)
```

Figure 45. Using the ADMMNICK specification to define a nickname for a Family 3 printer

After you create your nickname in TSO or native z/OS batch, a ddname ADMLIST is created. You can then send the formatted file to the printer you have chosen.

CICS: To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one in below.

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S,TONAME=370P (CICS)
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S (CMS)
```

Figure 46. Using the ADMMNICK specification to define a nickname for a Family 3 printer

After you create the ADMMNICK specification in CICS, link the name with a physical device by updating the DCT, as shown in the example in Figure 50 on page 191. Make sure TONAME in the ADMMNICK specification and DESTID in the DCT have matching values.

Example nickname for a family 4 GDDM printer on native z/OS batch, TSO or ISPF

To define the nickname 3900PRNT for a Family 4 GDDM printer, you might use an ADMMNICK specification similar to the one below.

```
ADMMNICK NAME=3900PRNT,TOFAM=4,DEVTOK=R87S
```

Figure 47. Using the ADMMNICK specification to define a nickname for a Family 4 printer

After you create your nickname, the ddname ADMIMAGE is created. You can spool the file to PSF/OS/390 and z/OS automatically through JES if you have the CSPOOL processing option set. For more information about Family 4 printing, see *GDDM System Customization and Administration*.

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, when you enter:

```
PRINT REPORT (PRINTER=MYPRINT2
```

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 ddname `MYPRINT2`.

Examples of nickname definitions

This section shows examples of nicknames you might use for Family 1, 2, or 3 devices. For examples on defining nicknames for Family 4 devices, see *GDDM System Customization and Administration*.

- **3800, 3812, or 3820 printer, 6 lines per inch:** Use the following definition to define the nickname GDDMPRT1 for a Family 3 printer:

  ```
  GDDMPRT1 ADMMNICK TOFAM=3,DEVTOK=S3800N6,NAME=MYPRINT1
  ```

- **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,DEVTOK=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=((PRINTCLTL,)),NAME=MYPRINT5
  ```

  The PROCOPT parameter specifies processing options using a print control (PRINTCTL) keyword, which allows you to 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 margins. The zero in this example suppresses page headings.

  Attention: If the print data set has RECFM=F, GDDM printing changes the DCB of the data set from RECFM=F to RECFM=V.

  For a list of print control options and how to use them, see *GDDM System Customization and Administration*.

- **A PC printer using GDDM-PCLK (for DOS users):** Use the following definition to define the nickname PCPRINT for a Family 1 printer:

  ```
  GDDMPRT6 ADMMNICK TOFAM=1,FAM=0,NAME=PCPRINT,TONAME=*,ADMPCPRT
  ```

  where * indicates the user's current device or the default value.

  To print to a workstation printer connected to DOS, GDDM-PCLK must be installed on your workstation.

Updating the GDDM defaults module with the nickname

Use this information to update the GDDM defaults module on native z/OS batch, TSO, and CICS.

Native z/OS batch, TSO and ISPF: In TSO and native z/OS batch, the external defaults module is ADMADFT.

The default modules also contain default values for the GDDM product. The modules are stored as members of the SADMSAM data set.

To update the modules with your nickname specification:

1. Edit the source file to add the nickname.
2. Enter your ADMMNICK specification after the ADMMDFT statements in the module.
3. Reassemble and link-edit the changed default module.

For more information on the defaults modules, see *GDDM System Customization and Administration*.

CICS: In CICS, the ADMMNICK nickname specifications reside in the GDDM external defaults module ADMADFC, which is supplied with the GDDM product.

The default modules also contain default values for the GDDM product. The modules are stored as members of the SADMSAM data set.

To update the modules with your nickname specification:

1. Edit the source file to add the nickname.
2. Enter your ADMMNICK specification after the ADMMDFT statements in the module.
3. Reassemble and link-edit the changed default module.

For more information on the defaults modules, see *GDDM System Customization and Administration*.

Testing the nickname definitions in external default files for native z/OS batch, TSO, and ISPF

Test your nickname definitions by placing them in an external default file and printing with them until you are satisfied they are working correctly. Then you can assemble them into external default modules.

GDDM uses external default modules more efficiently than a data set to find a given nickname.

The decision to use external default files or modules affects a user's JCL, because an external default file requires a DD statement, while an external default module must be a member of a STEPLIB library. Your GDDM administrator can advise you on the JCL changes.

Allocating the nickname file for native z/OS batch, TSO, and ISPF

For TSO and 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:

```
//ADMDEFS DD DSN=LOCAL.GDDM.NICKNAME,DISP=SHR
```

Using nicknames in 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 a physical device it manages.

Linking a Family 2 nickname with a physical 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 described to CICS using VTAM and TCT entries, you must describe the printer as queued (GDDM Family 2 device). When using a Family 2 device, your ADMMNICK specification for TONAME points to a CICS TCT entry, as opposed to a DCT entry for Family 3 devices.

For example, for this nickname specification:

```
ADMMNICK NAME=GRAPHIC,TOFAM=2,DEVTOK=R87S,TONAME=GRAP
```

you can update the CICS TCT using a macro similar to the example shown below.

```
GRAP     DFHTCT TYPE=TERMINAL,
              ACCMETH=VTAM,
              TRMIDNT=GRAP,
              TRMTYPE=SCSPRT,
                  . . .
                  . . .
                  . . .
```

Figure 48. Defining to CICS a nickname for a Family 2 GDDM printer

Linking a family 3 nickname with a physical device: To use Family 3 devices, set up a GDDM nickname table as shown below.

```
GDDMPRT ADMMNICK TOFAM=3,   FAMILY (SYSTEM PRINTER)           X
           NAME=SYSPRT,     PRINTER NAME (NICKNAME)            X
           DEVTOK=S1403W6,  DEVICE TOKEN (1403)                X
           TONAME=SYSP      TONAME MUST MATCH CICS DCT ENTRY
```

Figure 49. Defining to CICS a nickname for a Family 3 GDDM printer

GDDM System Customization and Administration describes the process of incorporating the nicknames into the user default specifications and assembling the user default specifications into external defaults module ADMADFC.

The TONAME parameter must have a matching entry in the CICS DCT as shown in Figure 50 on page 191.

```
* 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 50. Adding a TONAME entry to the CICS DCT

You also need to add the ddname ADMSYSP to the CICS start-up JCL, as follows:

```
//ADMSYSP DD SYSOUT=A
```

Add the TYPE=SDSCI entry shown in Figure 50 after all other TYPE=SDSCI entries in the DCT. The device address (SYS097) corresponds to the printer, 04E, according to the assign statement in the startup JCL. If you use SYSLST, CICS STATS is part of your QMF report. Instead, use an alternate printer.

Using QMF services to handle printing

Use this information for handling printing on native z/OS batch, TSO, and CICS.

Using 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 to print a report, table, SQL or QBE query, procedure, or the 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 using either the following JCL:

```
//DSQPRINT DD DSN=&SYSUID..PRINT.DATA,DISP=MOD
```

or the following CLIST:

```
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 using the following syntax:

```
//DSQPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
```

Set the CC option to YES.

If you are using ISPF: You can use the QMF-supplied DPRE (Display Printed Report) command synonym to view the effects of the width and length values you specified without having to print the report. This is applicable only while using DSQPRINT.

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 are limited to 32, 767 rows of output. They route data only to local print destinations. If you use temporary storage, you need to write a program that routes the data from the queue to the transient data queue, or view the report online with the CICS-supplied transaction CEBR.

CICS transient data queues are limited only by the amount of storage associated with the CICS DCT 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. Some intrapartition data queues are limited to 32,767 rows.

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 a value on the PRINT command, QMF uses values stored in the global variables DSQAP_CICS_PQNAME and DSQAP_CICS_PQTYPE.

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 terminal uses the same queue name, appending the previous report.

Viewing a report from a CICS temporary storage queue

You can view a report with the CICS-supplied transaction CEBR.

Defining a synonym for the print function key

Use these instructions to define a synonym in native z/OS batch, TSO, and CICS.

Native z/OS batch, TSO and ISPF

You can customize your system so you can print an object without exiting QMF. You can use a local print utility by simply pressing the Print function key, if you define a command synonym for printing and customize your Print function key.

1. Create a REXX exec or CLIST to locally print the current object. Here is a sample, using the QMF callable interface:

```
/* 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 QMF users prefer to bypass the print command and simply export the object for local printing. In this case your exec looks something like:

```
/* PRTQMF REXX EXEC for local print utilities called DSPRINT */
CALL DSQCIX "EXPORT PROC TO MYPROC"
DSPRINT '&SYSUID..MYPROC.PROC'
```

2. Create a QMF command synonym for printing. Here is a sample query that creates a command synonym PRTQMF to execute the PRTQMF exec:

```
INSERT INTO COMMAND_SYNONYMS (VERB, SYNONYM_DEFINITION, REMARKS)
VALUES('PRTQMF','TSO PRTQMF','Print QMF Proc')
```

3. You can now customize a function key on the procedure panel to use this command synonym. You need to customize for each panel. A query to customize function key 4 on the procedure panel looks like this:

```
INSERT INTO PFKY__TABLE (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('PROC','K', 4, 'PRTQMF')
```

This example assumes that the user's profile has the PFKEYS column value set to PFKY_TABLE, the name of the function key customization table. (After running the query, QMF must be restarted to implement the function key change.)

Defining a synonym for the print function key for CICS

You can customize to allow a user to print an object (in the following example, a report) without exiting QMF.

Use this technique to invoke a local print utility when the Print function key is pressed.

1. Create a QMF procedure called PRT__QMF. This sends the object to temporary storage, then starts a transaction that prints the object.

```
PRINT REPORT (QUEUENAME=QMFREPT,QUEUETYPE=TS)
CICS QMFP (FROM='QMFREPT')
```

2. Create a QMF command synonym for printing. Here is a sample query that creates a command synonym PRTQMF to execute the PRTQMF exec:

```
INSERT INTO COMMAND__SYNONYMS (VERB, SYNONYM__DEFINITION, REMARKS)
VALUES('PRTQMF','RUN PRT_QMF','Print QMF Report')
```

3. You can now customize a function key on the report panel to use this command synonym. You need to customize a key for each 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')
```

This example assumes the user's profile has the PFKEYS column value set to PFKY_TABLE, the name of the function key customization table. (After the query runs, QMF must be restarted to implement the function key change.)

Printing objects

The rules for printing QMF and database objects vary, depending on the type of object. The table below summarizes the requirements for each object.

Table 45. Summary of print requirements for QMF and database objects

Object type	Nickname required	GDDM gets control	Where output is routed for z/OS
Chart	Yes	GDDM ICU always gets control when the PRINT command is issued.	Output is controlled by GDDM.
Form	Yes	GDDM always gets control when the PRINT command is issued.	Output is controlled by GDDM.
QBE query	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Procedure	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Profile	No	Only if the nickname is supplied on the PRINT command or in 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	Only if the nickname is supplied on PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
SQL query	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Table	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.

Chapter 16. Customizing QMF commands

QMF command synonyms help customize QMF commands by allowing you to define your own terms and link them to QMF, CICS on z/OS, or TSO commands. A synonym might be another word for a command, or it might be a term that does the work of several commands.

After you create a command synonym, QMF end users can enter the synonym on the command line in the same way they enter a QMF command.

Using the default synonyms provided with QMF

QMF provides four applications that can be used as installation-defined commands. After installation, these synonyms appear in the Q.COMMAND_SYNONYMS table. Users with access to this table can call these applications by entering the appropriate synonym, as if it were a QMF command.

Workstation database server users: After installation, these synonyms appear in the table Q.COMMAND_SYN_TSO.

Default synonyms on z/OS

Display printed report
Synonym is DPRE. Displays the user's current report in its printed form.

Batch query/procedure
Synonym is BATCH. Allows the user to run a query or procedure in batch mode.

Layout Form
Synonym is LAYOUT. Lets the user tailor reports without running a query. For information on the Layout command syntax and an example of how to use this application, see *DB2 QMF Reference*.

Bridge to ISPF
Synonym is ISPF. Lets the user temporarily leave interactive mode QMF and "bridge" to an ISPF/PDF session. After the session is ended, the user returns to QMF at the point where the ISPF command was issued. For more on the ISPF application, see *UsingDB2 QMF* and *Developing DB2 QMF Applications*.

ISPF considerations
The synonyms DPRE, BATCH, LAYOUT, and ISPF are valid only if

QMF is started under ISPF. If QMF is not started under ISPF, you can access ISPF by entering TSO ISPSTART.

Displaying printed reports (DPRE) in ISPF

A printed report does not look exactly as it does on-screen. For example, the displayed report is treated as a single page even with one or more page breaks in the printed report.

The differences between the printed report and its displayed version are largely cosmetic; the facts and figures on the screen and those on the printed page are the same. However, the differences can be important. For more detailed information about the differences, see *Using DB2 QMF*. IBM supplies the QMF application DPRE to display the report as it will look when printed. After QMF is installed, the application can be invoked using a command stored in the Q.COMMAND_SYNONYMS table. The application is shared for everyone's use.

DPRE components under TSO are a procedure named Q.DSQAER1P in the database and a CLIST named DSQABR13 in the library QMF810.SDSQCLTE.

Using DPRE: To use DPRE, load the DATA object with the report data and the FORM object with the appropriate form, then issue the command:

```
DPRE
```

The application then generates the printer output and displays it through the ISPF browse facility. After you are done browsing, the printer output disappears.

If you are using an 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 or the translated online help for the command.

Report Parameters*:* The LENGTH parameter for the report being browsed is taken from PROFILE. The WIDTH parameter specified in PROFILE is used if it is less than 132 (lrecl); otherwise, a width of 132 (lrecl) is used because this is the length specified in the TSO allocate statement. If 132 is too small, the TSO allocate statement for DSQPRINT can be changed to accommodate a larger width.

Performance Considerations*:* The design of QMF encourages users to develop their printed reports by alternately modifying the FORM panels and displaying REPORT, until the report suits the user's needs. With DPRE, the user can alternate changing the FORM panel and browsing the tentative report with DPRE. Users should be aware, however, that this method of

development is more expensive than the first method, and should be used sparingly when resources are at a premium.

For a large report, all the rows of the report are fetched before the report is displayed.

Responding to Errors*:* DSQPRINT is the ddname for the data set that receives output from QMF PRINT commands in which PRINTER=' ' is either expressed or implied. When a user runs DPRE, DSQPRINT is redefined as the data set that holds the material to be browsed. If an error stops the run, this definition might still be in effect.

Customizing DPRE: Important: When making modifications to any file, first rename it and be sure to keep backup copies of original and modified files.

Under TSO, you can change two areas in DPRE:

- Handling the BROWSE data set

  The application reallocates DSQPRINT as a sequential data set created for the user. This data set contains the printed form of the report for the user to browse. You can change the name of this data set and its disposition.
- Modifying the function keys for DPRE

  To modify the function keys for DPRE, you must edit the QMF PROC Q.DSQAER1P and QMF810.SDSQCLTE(DSQABR13). For example, if you want to change the DPRE application function key 12 from CURSOR to RETRIEVE, you need to do both of the following:
  - In Q.DSQAER1P, change the value on the PF12CON line from CURSOR to RETRIEVE.
  - In the CLIST DSQABR13, change the value for both ZPF12 and ZPF24 from CURSOR to RETRIEVE.
- Reallocating DSQPRINT

  After a user finishes browsing the report, DSQPRINT must be reallocated to what it was before the application was called. The following statements in the application do this for you. They are in the procedure DSQAER1P.

  ```
  ADDRESS TSO "ATTR DSQDPRA LRECL(133) RECFM(F B A) BLKSIZE(1330)"
  ADDRESS TSO "ALLOC    DDNAME(DSQPRINT) SYSOUT(A) USING(DSQDPRA)"
  ```

  You can change the ALLOC statement. For example, you can change the output class for DSQPRINT from A to C. You might want to do this if output class C handles confidential printouts and most QMF reports at your installation are confidential. The modified ALLOC statement looks like this:

  ```
  ADDRESS TSO "ALLOC    DDNAME(DSQPRINT) SYSOUT(C) USING(DSQDPRA)"
  ```

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. When you enter a command, QMF first checks the synonym table for a match. If there is no match, QMF assumes the command is a base QMF command. When you enter the letters *QMF* in front of any command, QMF automatically assumes the command is a base QMF command and does not check the synonym table for a match.

Creating a command synonym table on z/OS

Use the following procedure to create a command synonym table. Then see "Entering command synonym definitions into the table" on page 202 for instructions on entering your synonyms and their definitions.

1. If necessary, acquire or add a table space to hold the command synonym table. On z/OS, the storage container for a table is referred to as a table space. The examples below use the z/OS default table space name, TBSPACE1. If you do not have an available table space, create one for your table with a query like the following:

```
CREATE TABLESPACE DSQTSSN1
    IN DSQDBCTL
       USING STOGROUP DSQSGSYN
          PRIQTY 100
          SECQTY 20
          LOCKSIZE PAGE
          BUFFERPOOL BP0
          CLOSE NO
```

Figure 51. Creating a table space

   Running this query creates the table space DSQTSSN1. The storage group and database for this table space are also those for the table space containing Q.COMMAND_SYNONYMS.

   You might be able to use DSQDBCTL.DSQTSSYN as a table space. The Q.COMMAND_SYNONYMS table resides in DSQDBCTL.DSQTSSYN.

2. From the QMF SQL query panel, run an SQL CREATE TABLE statement similar to the one in Figure 52 on page 201 to create the table. Substitute your own table name in place of COMMAND_SYNONYMS and your own table space name for TBSPACE1. Type the other portions of the query 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 TBSPACE1
```

Figure 52. 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 users know what function each synonym performs.

3. Add comments to the DB2 system catalog using the following example for the COMMAND_SYNONYMS table created with the query in Figure 52.

   ```
   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, but if you do, one comment might be about the table, others might 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:

   ```
   COMMENT ON COLUMN COMMAND_SYNONYMS.AUTHID
     IS 'PRIVATE SYNONYM: USE AUTH ID. PUBLIC SYNONYM: USE NULL'
   ```

   By running a subsequent COMMENT ON query, you can replace the current one. For more on COMMENT ON queries, see the *DB2 UDB for z/OS Administration Guide.*

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 displays a warning message on the QMF Home panel after initialization.

Entering command synonym definitions into the table

After you create a command synonym table, use an SQL INSERT statement similar to the one in Figure 53 to enter your synonyms into the table. You can also use the Table Editor to update the table, as explained in *Using DB2 QMF*.

```
INSERT INTO COMMAND_SYNONYMS (VERB,OBJECT,SYNONYM_DEFINITION)
  VALUES('COMPUTE', 'MONTHLY_SALES', 'RUN PROC JONES.SALES_FIGURES')
```

Figure 53. Creating a command synonym definition

After it is activated according to the procedure in "Activating the synonyms" on page 208, the synonym COMPUTE MONTHLY_SALES runs a QMF linear procedure called SALES_FIGURES, owned by user JONES.

The query in Figure 54 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 54. 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 Figures 53 and 54 follow guidelines that allow QMF to process each synonym correctly. The rest of this section explains these guidelines, which you need to follow to ensure that QMF correctly processes your entries for the VERB, OBJECT, and SYNONYM_DEFINITION columns in the table.

Choosing a verb

Every command synonym definition must have a verb. Only the object name is optional.

The verb is your own word for the QMF RUN command, CICS or TSO command 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 only procedures that return financial results.

Rules for the VERB column

Ensure entries in the VERB column of the synonym table:

- 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, and displays a warning message.

Valid Verbs:
Invalid Verbs:

COMPUTE
DO SALES (Blanks not allowed unless surrounded by double quotes)
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, instruct users to precede the command with the letters *QMF* when they 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. Using base verbs in verb-object synonyms has a similar impact.

Some base QMF commands must be followed by a parameter. 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. For other base commands you use, see the syntax diagrams in *DB2 QMF Reference* to find out if the command requires a parameter.

z/OS concerns

The verb is your own word for the QMF RUN command, CICS, or TSO command 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 only procedures that return financial results.

Choosing an object name

An object name is optional in a command synonym. When you do use an object name, however, ensure users specify both the verb and the object name; otherwise, QMF cannot find a match in the synonym table. Entries in the OBJECT column 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 quotes if the object name has blanks or other special characters. (Both QMF and the database manager remove the double quotes when the name is processed.)

The following examples show valid and invalid objects.

Valid Objects:
Invalid Objects:

PFKEYS
80CAT (first character is numeric)

MONTH_2_REPORT
ADJ%AGE (% not allowed)

“Net Sales”
JONES GROSS (double quotes required for blanks)

If you are using fully qualified table names: Object names can look like fully qualified table names; this is consistent with the QMF language. However, QMF objects other than tables cannot be referenced by three-part names. For example, the object name in the following QMF command has a fully qualified table name:

```
DISPLAY FORM.BACKUP
```

Choosing the synonym definition

The synonym definition is the QMF command or procedure that runs when the user enters the command synonym.

Choosing the synonym on z/OS

An entry in the SYNONYM_DEFINITION column can include:

- A RUN command that calls a QMF procedure or query. For example, RUN PROC JONES.SALES_DATA might be a synonym definition for the command synonym COMPUTE MONTHLY_SALES.
- A TSO command that calls a CLIST.

- A CICS command that starts another CICS transaction.

Your synonym definition can even include both types of commands if the definition runs a QMF linear procedure.

For information about developing complex applications to run in a command synonym, see *Developing DB2 QMF Applications*.

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, the procedure in Figure 55 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 TSO virtual storage or a CICS temporary queue. In Figure 55, XYZ is the name of the temporary storage queue.
3. Runs a CICS or TSO procedure to route the report from virtual storage to a predefined print destination. In Figure 55, RPTX is the transaction name. It runs asynchronously with QMF to route output to a destination named REPORTX.

```
-- Procedure name: SALES_PROC
 RUN QUERY SALES_DATA
 PRINT REPORT (QUEUENAME=XYZ,QUEUETYPE=TS)
 TSO RPTX (FROM=('REPORTX, XYZ'))
```

Figure 55. Sample procedure to run using a command synonym

Your definition for a synonym that runs this procedure might look similar to the one in Figure 56:

```
SYNONYM
VERB           OBJECT           DEFINITION
-----------    ---------------  -------------------
SHOW           SALES            RUN PROC SALES_PROC
```

Figure 56. Using a command synonym to run a linear procedure

If you are using an NLF: Make sure that the QMF commands in the queries, forms, and other objects included in the procedure are translated before you

use the command synonym that calls the procedure. Also ensure these components are suitable for the NLF you are using. Unless your procedure sets the DSQEC_NLFCMD_LANG variable to 1, ensure the commands are translated before you use the command synonym.

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, Figure 57 shows a definition that passes the value Q.STAFF for the table name, which is evaluated when MYQUERY runs.

```
SYNONYM
VERB        OBJECT     DEFINITION
----------  ---------  ------------------------------------
EXECUTE     -          RUN QUERY MYQUERY (&&TABLENAME=Q.STAFF
```

Figure 57. Using variables in command synonym definitions

MYQUERY might look something like:

```
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.

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 information is located within the synonym definition.

The synonym definition in Figure 58 shows an example of a synonym defined using &ALL.

```
SYNONYM
VERB        OBJECT     DEFINITION
----------  ---------  ------------------------
SHOW_INFO   -          RUN QUERY STAFFQUERY (&ALL)
```

Figure 58. Using the variable &ALL in a command synonym definition

The query named STAFFQUERY might look something like the following:

```
SELECT * FROM Q.STAFF
WHERE DEPT=&DEPT and JOB=&EMPLOYEE_JOB
```

After activating the SHOW_INFO synonym defined in the preceding example, 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'
```

Rules for &ALL: When you use the variable &ALL in a synonym definition:

- Use &ALL only once in a synonym definition.
- Always write &ALL in uppercase.
- Never follow &ALL with a number or letter.
- Any value you substitute for &ALL must be syntactically correct when QMF evaluates the entire command. For more information on syntax of QMF commands, see *DB2 QMF Reference*.

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 58 on page 206, 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: Follow these rules when keying your synonym definitions into the synonym table:

- Add single quotes around a 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, &ALL has single quotes in this synonym definition:

  ```
  RUN MYQUERY (&&NAMEVALUE='&ALL'
  ```

  If you search for the name O'BRIEN, you do not need to enter `'O'BRIEN'`, because QMF does this for you.
- Enter base verbs and keywords in uppercase.

  Literal information in the synonym definition is not converted to uppercase.
- 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 that 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 uppercase and the synonym definition is in lowercase, QMF cannot find the synonym definition that

matches the synonym the user entered. The CASE value of the user's QMF profile controls whether input is converted to uppercase. Use the SET PROFILE command to change the CASE value. This command is explained in *DB2 QMF Reference*.

Activating the synonyms

Command synonyms follow the same rules for abbreviation as QMF commands. 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. See *DB2 QMF Reference* for the proper abbreviations for QMF commands.

Activating the synonyms on z/OS

QMF Version 8.1 allows for the use of long names in the command synonym table. The Q.PROFILES column, SYNONYMS, is now VARCHAR(261) to allow for a 128 byte table owner ID and table name.

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.

   For example, 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, use the query in Figure 59:

```
Base QMF (English)
        German NLF
UPDATE Q.PROFILES
        UPDATE Q.PROFILES
SET SYNONYMS='COMMAND__SYNONYMS'
        SET SYNONYMS='GUMMOW.XYZ'
WHERE CREATOR='JONES'
        WHERE CREATOR='SCHMIDT'
AND TRANSLATION='ENGLISH'
        AND TRANSLATION='DEUTSCH'
AND ENVIRONMENT='TSO'
        AND ENVIRONMENT='TSO'
```

Figure 59. Activating a user's QMF command synonyms

Important: Always specify a value for TRANSLATION when you are updating Q.PROFILES, or you might change more rows than you intend.

The query in Figure 59 on page 208 applies to users who are already enrolled in QMF. You can use a similar query to update the SYSTEM profile. If you are enrolling a new user, use an INSERT query.

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 synonym table rather than the table itself, grant SELECT on only the view to prevent users from accessing synonyms not meant for their use. Views are discussed in "Minimizing maintenance of command synonym tables."
3. Instruct users to end the current QMF session and start another to activate the new synonyms.

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 monitor the table's statistics, refer to the *DB2 UDB for z/OS Administration* Guide.

To minimize the time you spend maintaining users' command synonym tables, consider either assigning one synonym table to all users or assigning a variety of different views of the same table. Both methods are discussed in this section.

Assigning one synonym 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. The query in Figure 60 assigns to every user of base (English) QMF a table named COMMAND__SYNONYMS.

```
UPDATE Q.PROFILES
  SET SYNONYMS='Q.COMMAND__SYNONYMS'
  WHERE TRANSLATION='ENGLISH' and ENVIRONMENT='TSO'
```

Figure 60. Assigning a single command synonym table to all QMF users

Assigning views of a synonym 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 synonym 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):

1. Add an AUTHID column to the synonym 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 query similar to that in Figure 61 to create a view on the synonym table. This query 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 61. 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.

The synonym table 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.

1. Add an AUTHID column to the synonym table if it does not have one.
2. Use a query similar to the one in Figure 62 to create the view on the synonym table. The example in the figure 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 62. 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 in 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 query similar to the one in Figure 63 to implement this method of maintaining command synonyms. The query 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 63. Creating a view that uses an extra table to control use of synonyms

Chapter 17. Customizing QMF 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 executes when a user presses the key.

Choosing the keys that you want to customize

QMF function keys appear on two types of panels: primary panels, which are full-screen panels such as FORM.MAIN and REPORT, and secondary panels, which appear as window dialog panels. Help, prompt, and Prompted Query panels are examples of secondary panels.

The tables in "Default keys on full-screen panels" show the default QMF function key labels and commands for both full-screen and window panels; use them to decide which function keys you want to change.

You cannot customize function keys on Table Editor panels. On other panels, you can choose QMF or installation-defined commands to associate with any function key label you modify.

Default keys on full-screen panels

Key	Executed command
Backward	BACKWARD
Cancel	CANCEL
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

Key	Executed command
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
Specify View	SPECIFY VIEW

Default keys on window panels

Key	Executed command
Attribute	SPECIFY ATTRIBUTES
Backward	BACKWARD
Cancel	CANCEL
Clear	CLEAR
Command	SHOW COMMAND
Comments	SWITCH COMMENTS

Key	Executed command
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
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.

For more information on the commands associated with these function keys, see *DB2 QMF Reference*.

Creating the function key table

Use these instructions to create the function key tables on z/OS.

Creating the table on z/OS

After you decide which function keys you want to customize, follow these steps to create a table that links your customized function key definitions with the appropriate panels:

1. Use an SQL CREATE TABLE statement similar to the one shown in Figure 64 to create the table. Substitute your own name for MY__PFKEYS. Under TSO, substitute your own table space for TBSPACE1.

```
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 TBSPACE1
```

Figure 64. Creating a function key table

   See the *DB2 UDB for z/OS Administration Guide* for information on creating a new table space.

2. Add comments to the DB2 system catalog using an SQL statement similar to the following:

   ```
   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. For more information on adding comments to the system catalog, see the *DB2 UDB for z/OS Administration Guide*.

   You do not need to add comments about your new table to the DB2 system catalog, but if you do, one comment might be about the table; 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:

   ```
   COMMENT ON COLUMN MY__PFKEYS.AUTHID
     IS 'PRIVATE PFKEY: USE AUTH ID. PUBLIC PFKEY: USE NULL'
   ```

   By running a subsequent COMMENT ON query, you can replace the current one. For more on COMMENT ON queries, see *DB2 UDB for z/OS SQL Reference*.

3. Create an index using an SQL statement similar to the following:

   ```
   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 displays warning messages on the Home panel if it finds more than one key definition for the same key, and writes information about the warning 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 customized 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 label text that appears on the screen.

Enter both rows for each key 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.

The next two sections discuss the values you need to enter for each row.

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 your customized function keys also work this way, make sure one of the two rows you enter into the table has the values shown in Table 46 on page 218.

Table 46. Values to customize your function key table

Column	Value	Information
PANEL	ID of the QMF panel you're customizing	"Full-screen panel identifiers" on page 221 shows the IDs you need to use for full-screen panels. "Window panel identifiers" on page 222 shows the IDs you need to use for specific window panels. 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 Specify panel of a Forms window, use the panel ID FOSPEC if you want the Specify 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. Changes 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 QMF issues when the key is pressed.
NUMBER	Number of the function key you're customizing	If you are changing the definition for F5, enter a 5 in this column.
PF__SETTING	Text of the command that runs when the key is pressed	Make sure this command is appropriate for the panel on which it appears. For example, the ENLARGE command is appropriate for only the QUERY 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 your end users need them. Enter the command in uppercase, because QMF does not convert terminal input to uppercase when it retrieves the commands associated with function keys. The command won't run if this value is lowercase and the CASE field of the user's profile has the value UPPER. Ensure that each panel 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.

If you are using an NLF: Make sure the underlying command has the correct national language translation; additionally, it is helpful if the label text for each key is written in the language of the NLF you are using.

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 the label appears on the screen, you need to add a second row to the table. In this row, make sure the columns of the function key table have the following values:

Table 47. Values to label your function key table

Column	Value	Information
PANEL	ID of the QMF panel you're customizing	This is the same ID you used for the first row of the definition, explained in "Linking a command with a function key" on page 217.
ENTRY_TYPE	L	L indicates that the row defines the label associated with the function key.
NUMBER	Number of the row where the key appears on the display, if you are customizing a full-screen panel	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 in "Linking a command with a function key" on page 217). For example, on the Home panel, F5 appears in row 1 and F12 appears in row 2.
PF_SETTING	Text of the function key labels	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 fill in the default key settings for keys you choose not to customize. See Figure 65 on page 220 for an example. For window panels, you need to type only the label of the key in this column. See Figure 66 on page 221 and Figure 67 on page 221 for examples.

Examples of key definitions

Use the examples in this section 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 in "Identifying the panel that you want to customize" on page 221. Use these tables to get the proper values for the PANEL column of the function key table.

Important: Ensure that each customized secondary panel has a key set to the CANCEL command to enable the user to exit the panel.

Entering a definition for a key on a full-screen panel

Use the SQL queries shown in Figure 65 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 query 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 65. Changing a function key for a QMF command on the Home panel

The QMF Home panel now displays Import for F2:

```
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 ===>
```

In the PF__SETTING column of the second query, 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 query, the Home panel looks like this:

```
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 ===>
```

Entering a definition for a key on a window panel

The SQL queries in Figure 66 on page 221 add a 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 66. Changing a function key on the Specify panel of Prompted Query

Entering a key definition for a Help or prompt panel

The SQL queries in Figure 67 add a 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 67. 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 changes you make to one help or prompt panel are appropriate for all the help and prompt panels in that class.

Identifying the panel that you want to customize

Use the tables in this section to help you determine what ID to enter in the PANEL column of your function key table. The panel ID appears in the upper left corner of the panel, when the global variable DSQDC__SHOW__PANID is set to 1, using the following command:

```
SET GLOBAL (DSQDC__SHOW__PANID=1
```

Full-screen panel identifiers

The full-screen panel identifiers for the QMF English base are listed in Figure 68 on page 222. For the list of valid full-screen panel IDs in a QMF NLF, type in the QMF command: HELP DSQ22957 from within any panel of the QMF NLF. Valid full-screen panel IDs for each QMF NLF are listed in the language-specific versions of the DSQ22957 message. Enter the identifiers in the PANEL column of the function key table exactly as they are shown here or in the message text.

```
PROMPTED QUERY      FORM.BREAK1     FORM.COLUMNS
SQL QUERY           FORM.BREAK2     FORM.CONDITIONS
QBE QUERY           FORM.BREAK3     FORM.DETAIL
PROC                FORM.BREAK4     FORM.FINAL
PROFILE             FORM.BREAK5     FORM.MAIN
REPORT              FORM.BREAK6     FORM.OPTIONS
GLOBALS             FORM.CALC       FORM.PAGE
HOME
```

Figure 68. Full-screen panel identifiers for QMF English base

Window panel identifiers

Use the tables in this section to reference window panel IDs. If you set the global variable DSQDC__SHOW__PANID to display the panel IDs, you will notice that each ID shown in these tables is prefaced by 4 characters when it appears on the screen.

Window panels not named in the tables do not have unique panel IDs, and can be customized 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 actually part of the ID.

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
GLSHVA	Show Variables
GLXXXX	Global Variables Window Class

Help and prompt windows

Panel identifier	Title or description
HEXXXX	Help Window Class
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
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
QPEXPR	Expression
QPJOCO	Join Columns
QPJOTA	Join Tables
QPROBE	Rows - Between
QPROCH	Rows - Change (left side)

Panel identifier	Title or description
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
QPXXXX	PQ Window Class

Activating new function key definitions

QMF Version 8.1 allows for the use of long names in the function key table. The Q.PROFILES column, PFKEYS, is now VARCHAR(261)to allow for a 128 byte table owner ID and table name.

Activating definitions on z/OS

To enable users to use the customized function key definitions you created:

1. Update the PFKEYS field of the user's profile with the name of your function key definitions table.

   For example, use a query like the one in Figure 69 on page 225 to assign to English QMF user JONES the table MY__PFKEYS, and to German NLF user SCHMIDT the table MEIN__FKY. Always include a value for the TRANSLATION and ENVIRONMENT columns in a query that updates the Q.PROFILES table.

```
Base QMF (English)
        German NLF
UPDATE Q.PROFILES
        UPDATE Q.PROFILES
SET PFKEYS = 'MY__PFKEYS'
        SET PFKEYS = 'MEIN__PFKY'
WHERE CREATOR='JONES'
        WHERE CREATOR='SCHMIDT'
AND TRANSLATION = 'ENGLISH'
        AND TRANSLATION = 'DEUTSCH'
AND ENVIRONMENT = 'TSO')
        AND ENVIRONMENT = 'TSO')
```

Figure 69. Making customized function keys accessible to a user on OS/390

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.

   The procedures for assigning views of a function key table are the same as those for command synonym tables, discussed in "Minimizing maintenance of command synonym tables" on page 209. Use the strategies discussed in that section to decide whether to assign a table or a view to individual users or groups of users.
3. Instruct users to end the current QMF session and start another to activate the new function keys.

Testing and problem diagnosis for the function key table

After you have activated the new function key definition by inserting the function key table name into your Q.PROFILES entry, the new definitions are ready to be tested. The new definitions do not take effect until one of two conditions is met.

- You close QMF and then start a new QMF session.
- From within QMF, you reconnect to QMF by entering the CONNECT TO *locname* command, where *locname* is the same location name that you see on the QMF home panel.

If you see the message ″Warning messages have been generated″ after you perform one of these two actions, exit QMF and examine your QMF trace data (DSQDEBUG) output. The trace provides messages that you can use to fix problems. If you do not see the new function key definitions after reconnecting to QMF, it is possible that the Q.SYSTEM_INI proc or other user controlled features is covering up a possible ″Warning messages have been generated″ message. In this case, exit QMF and review the DSQDEBUG trace output.

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 your Q.PROFILES row entry.

Chapter 18. Creating your own edit codes for QMF forms

Note: This chapter contains General Use Programming Interface and Associated Guidance information.

QMF forms

QMF forms help users to control the format of data returned from the database. Use edit codes in the EDIT column of the MAIN and COLUMNS panels of the QMF form to format report data in different ways. For example, 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.

In DB2 QMF Version 8.1, a new edit code 'M' has been created. Edit code 'M' represents that metadata will be shown and that the Descriptor Area (DA) will be displayed in character format rather than actual column data. The LOB data types CLOB, DBCLOB, BLOB, and the defined length field are displayed by default for LOB columns. If a user wants to view the actual LOB data, he can modify FORM.MAIN or FORM.COLUMN, and change the column edit code to 'C' or 'CW' to display character data.

If the edit codes supplied with QMF do not meet the report editing needs of your site, you can use the information in this chapter to create your own edit codes to be used in the EDIT column of the FORM.MAIN and FORM.COLUMNS panels. *DB2 QMF Reference* shows the edit codes supplied with QMF.

This chapter also shows how to write an edit exit routine in High Level Assembler, PL/I, or COBOL to format the data described by your edit code. QMF provides both a standard interface to your edit exit routine and a sample edit exit program you can use as a starting point for writing your own.

QMF supports edit routines in 31-bit or 24-bit AMODE or RMODE; however, some versions of supported languages do not support 31-bit addressing. QMF running in CICS requires 31-bit addressing.

Choosing an edit code

Create either a Uxxxx or Vxxxx edit code to be handled by your edit exit routine. For U codes, data passed to the edit routine has the internal database representation of the source 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 indicate processing for either character or numeric data. U and V must be uppercase. Replace *xxxx* with zero to four characters (letters, digits, or special characters) that can be entered from a terminal; 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
```

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 data type is extended floating point, ensure that the programming language supports it. For example, VS COBOL II does not handle extended floating point data. In this case, use V codes.

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 Table 48.

Table 48. How QMF converts numeric data according to data type

If data type of original numeric data is:	**QMF converts it to this length:**
Small integer	5
Integer	11
Decimal	Equal to the precision of the original data (raised to an odd number if the original data is even)
Floating point	15 or more depending on the base 10 exponent
Extended floating point	30 or more depending on the base 10 exponent

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 terminal to uppercase, and the edit code might not be recognized. If your edit code is written to accept edit codes in mixed case, enter the edit codes when case is set to mixed.

Handling DATE, TIME, and TIMESTAMP information

You can also use the edit code exit to format date, time, and timestamp values.

If your installation supports date/time data types, you can format columns with data types of DATE, TIME, and TIMESTAMP. This enables your users to use local date/time exit routines. For more information about these data types, see *Using DB2 QMF*. You need to remember that these are DB2, not QMF exits. For details about how these exits are created refer to the appropriate *DB2 System Administration*.

Your edit routine can format data from these columns, just as it can format data from columns of the 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 format of the string is described in Table 49.

Table 49. Formatting DATE, TIME, and TIMESTAMP data

Data type	Form of the string
DATE data	yyyy-mm-dd where: **yyyy** Specifies the year. It is always a four-digit number. **mm** Specifies the month (01 for January, ... 12 for December). It is always a two-digit number that can contain a leading zero. **dd** Specifies the day of the month. It is always a two-digit number that can contain a leading zero. The dashes (-) represent true dashes. For example, 1990-12-12 is the date December 12, 1990.
TIME data	hh.mm.ss where: **hh** 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. **mm** Specifies the minute. It is always a two-digit number that can contain a leading zero. **ss** Specifies the second. It is always a two-digit number that can contain a leading zero. The periods represent true periods. For example, 13.08.36 is 1:08 P.M. and 36 seconds in the notation commonly used in the United States.

Table 49. Formatting DATE, TIME, and TIMESTAMP data (continued)

Data type	Form of the string
TIMESTAMP data	yyyy-mm-dd-hh.mm.ss.nnnnnn where: **yyyy-mm-dd** Specifies the date in the same way it does for DATE data. **hh.mm.ss** Specifies the time of day in the same way it does for TIME data. **nnnnnn** Specifies a six-digit number that extends the count of seconds (ss) down to the nearest microsecond. For example, 1990-12-12-13.08.36.123456 is 1:08 P.M. and 36.123456 seconds on December 12, 1990, in the notation commonly used in the United States.

For the data types available, see the ECSINTYP field in Table 50 on page 232.

Calling your exit routine to format the data

Use these instructions to call your exit routine on z/OS.

Calling your exit routine on z/OS

Figure 70 shows how QMF and your edit exit routine work together to format data using the edit codes you define.

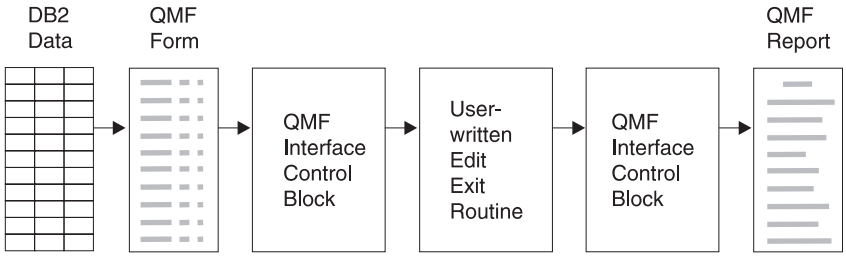


Figure 70. How a user edit routine works with QMF for TSO/CICS

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, which are discussed in "Fields of the interface control block" on page 232. QMF also passes into the input area the data to be formatted and an output area that holds the formatted result.

IBM supplies six different versions of a sample edit exit routine in QMF810.SDSQSAPE.

Language	TSO and native z/OS Batch	CICS
COBOL	DSQUXDTC	DSQUXCTC
PL/I	DSQUXDTP	DSQUXCTP
Assembler	DSQUXDTA	DSQUXCTA

The sample program supports two edit codes:

VSS Adds dashes to a social security number or a character string.

UDN Transforms a department number into its department name, using a table internal to the program.

The sample program is commented so you can more easily see how a user edit routine works. You can use the sample as a template for creating your own program. These routines can be found in QMF810.SDSQSAPE on z/OS.

QMF supplies the user edit routine DSQUEDIT for TSO and native z/OS, and a reentrant module, DSQUECIC, for CICS, which are located in the QMF library QMF810.SDSQLOAD. Delete or rename the QMF-supplied module when you are ready to use your edit routine.

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 database 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 takes the parameters in the following list:

- The interface control block.
- The value of ECSINPT, the data from the input area to be formatted.
- The value of ECSRSLT, the output area containing the formatted result. ECSRSLEN contains the amount of storage actually passed to this output area on each call. The result cannot be column wrapped.

  Important: Do not use more memory in the output area than is indicated in the ECSRSLEN field, or you will see QMF error DSQ60439 - User edit program memory overwrite.

User edit programs may require modifications. To correct this application error, do one of the following:

- Increase the WIDTH of the COLUMN by modifying the edit code in the FORM to the correct length expected on the report.
- Check the length of ECSRSLEN to determine if your program should PAD or TRUNCATE the results passed back to QMF.

ECSINPT, ECSRSLT, and ECSRSLEN are fields of the interface control block, explained in Table 50.

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 your edit exit routine is written in. These fields are shown in Table 50. Unless otherwise stated, each field relates to all formatting calls.

These same fields appear in each sample program (one for each programming language supported) shipped with QMF. You can include these field names in your own source program. The QMF production disk contains the sample programs.

Table 50. Fields of the QMF interface control block

Name	Contents
ECSDECPT	Contains the current decimal point symbol as determined by the DECOPT option of PROFILE (period or comma).
ECSECODE	Contains the user edit code.
ECSERRET	Contains a zero at the point of call. Set this to a nonzero return code to record an error. Use one of the values on the following list for an error of the indicated type: **Number** **Error** **99101** Unrecognized edit code **99102** Improper input data type for edit code **99103** Invalid input value for item to be formatted **99104** Item to be formatted is too short **99105** Not enough room for result in ECSRSLT (result is too wide for the space allotted) 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 backup help panel, is displayed.
ECSFREQ	Holds E for a formatting call, T for a termination call.
ECSINLEN	Contains the length, in bytes, of the value to be formatted.

Table 50. Fields of the QMF interface control block (continued)

Name	Contents
ECSINNUL	Holds an N if the value to be formatted is null.
ECSINPRC	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.
ECSINSCL	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.
ECSINSGN	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.
ECSINTYP	Indicates, in database terms, how the value to be formatted is represented. Applies to edit codes of every type. Values can be: 384 DATE data type 388 TIME data type 392 TIMESTAMP data type 448 VARCHAR data type 452 CHAR data type 456 LONG VARCHAR data type 464 VARGRAPHIC data type 468 GRAPHIC data type 472 LONG VARGRAPHIC data type 480 FLOAT data type 484 DECIMAL data type 496 INTEGER data type 500 SMALLINT data type 940 Extended floating point data type The extended floating point data type is not supported by the database (or by COBOL); it is limited to functions such as AVERAGE and STDEV. Extended floating point values are precise to more than 30 digits.
ECSNAME	Contains the name of the control block, which is DXEECS. Serves as an eye catcher in storage dumps.
ECSRQMF	Set this to T to request a termination call.
ECSRSLEN	Contains the length of the output area, in bytes. (Value is taken from the column WIDTH in the FORM)
ECSTHSEP	Contains the thousands separator as determined by the DECOPT option of PROFILE (blank or a comma).
ECSUSERS	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.

Fields that characterize the input area

Restriction: This section does not apply to values from DATE, TIME, and TIMESTAMP columns. For information on values for those types, see "Handling DATE, TIME, and TIMESTAMP information" on page 228.

During a session, the subprogram DSQUXDT might need to service many different edit codes. If it does, consider making your routine an executive routine, which does nothing but analyze the edit codes passed to it and then invokes an appropriate routine to do the actual formatting. The design makes the source code easier to read and easier to modify when new user edit codes are devised.

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 value to be formatted appears in the field ECSINPT. How it is represented depends on whether the value to be formatted 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, a data type not explicitly supported by DB2. 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, 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 field ECSINLEN. 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 field ECSINSGN, and the position of the decimal

point is in the field ECSINSCL. 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 string. For all character values, the string length (in bytes) is in the ECSINLEN field.

Fields that characterize the output area

The ECSRSLT field receives the formatted output in the form of a character string that completely fills the field. Upon input, this field is always blank. The length of this field (in bytes) is in the ECSRSLEN field. QMF blanks out ECSRSLT before calling the edit routine. The output area is temporary storage and can hold no more than 32,767 rows of output.

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 whenever 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.

Writing an edit routine in HLASM (high level assembler)

You can write an edit routine in Assembler for native z/OS, TSO, and CICS.

Writing an edit routine for native z/OS, TSO, or ISPF

The QMF edit exit interface for Assembler consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSA
- Control program, which is shipped with QMF as DSQUXIA
- Your edit exit program, which is named DSQUXDT

Figure 71 on page 236 shows the program structure of an Assembler edit exit routine for native z/OS, TSO, or ISPF.

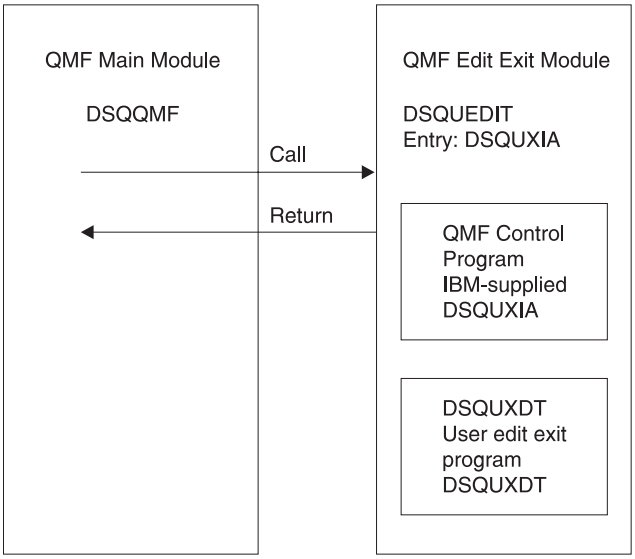


Figure 71. Program structure of an Assembler edit exit routine for TSO and native z/OS

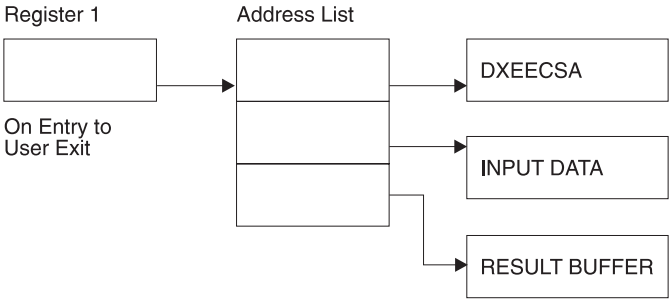
Example program DSQUXDTA

The IBM-supplied sample edit program for Assembler, DSQUXDTA, is located in the QMF810.SDSQSAPE library for z/OS. The sample program is commented so that you can modify it to suit your needs. If you plan to use this example program, copy it to your program library and change its name to DSQUXDT. Near the bottom of this file is a COPY statement for DXEECSA. which is a member of DSQUSERE MACLIB on z/OS. DXEECSA defines the input fields, giving them the names we are using in this chapter.

How an Assembler edit routine interacts with native z/OS

The user edit program is called as a subroutine in TSO 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.

An Assembler DSECT for DXEECS is shipped with QMF as DXEECSA, located in library QMF810.SDSQUSRE on z/OS. Include this DSECT in your program using the Assembler COPY statement.

Return control to QMF in the standard convention by restoring registers to their value 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 8, 9, and 10 through the 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.

It follows that registers 10, 9, and 8 point, respectively, at the control block, the value to be formatted, and the storage reserved for the formatted results.

Return control to QMF using the standard convention by restoring the registers to their value at the time of the call, and returning to the address in register 14.

How an Assembler edit routine interacts with QMF

The interface control block between QMF and the user edit interface DSQUXDT is DXEECS. It 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. This 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.

Assembling and link-editing your program on z/OS

During the Assembler, QMF edit exit interface control block DXEECSA, located in QMF sample library QMF810.SDSQUSRE in TSO or native z/OS, must be available in a macro library.

Create a new QMF edit exit module DSQUEDIT by including your edit program DSQUXDT with the IBM-supplied control module DSQUXIA, which is located in the QMF module library QMF810.SDSQLOAD. The IBM-supplied control module DSQUXIA must be specified as the entry point.

The 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. We recommend the 31-bit addressing mode.

Example statements for assembling and link-editing on z/OS

The following are example statements for assembling and link-editing your job for TSO or native z/OS:

```
//sampasm    JOB
//STEP1      EXEC PROC=ASMHCL
//* Provide Access to QMF Edit Macro DXEECSA
//C.SYSLIB   DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//C.SYSIN    DD  *
          .
          Your program or copy of QMF sample DSQUXDTA
          .
/*
//* Provide Access to QMF Interface Module
//L.QMFLOAD  DD DSN=QMF810.SDSQLOAD,DISP=SHR
//L.SYSIN    DD  *
          INCLUDE QMFLOAD(DSQUXIA)
          ENTRY DSQUXIA
          MODE  AMODE(31) RMODE(ANY)
          NAME  DSQUEDIT(R)
/*
```

Writing an edit routine in Assembler for CICS

The QMF edit exit interface for Assembler in CICS consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSA
- 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 DSQUECIC

Figure 72 shows the program structure of an Assembler edit exit routine for CICS.

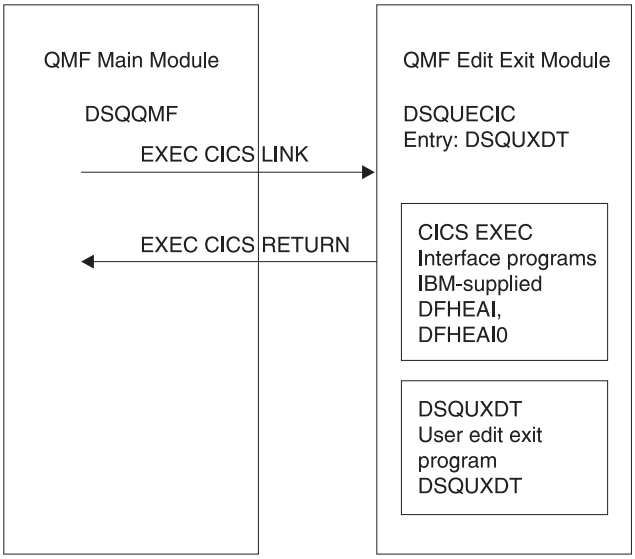
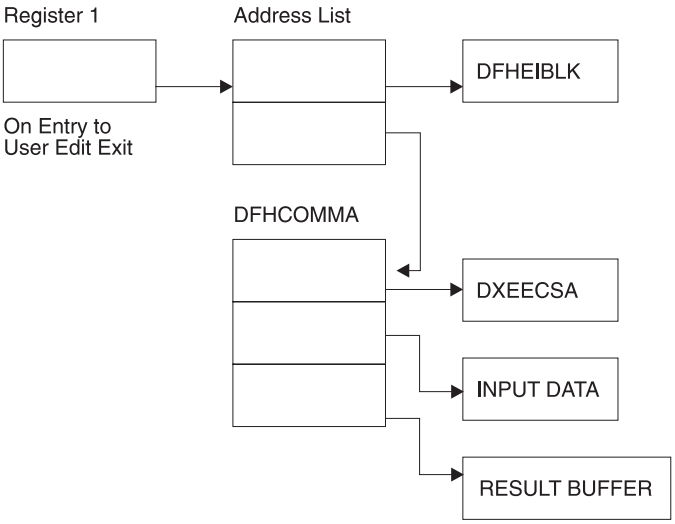


Figure 72. Program structure of an Assembler edit exit routine for CICS

How an Assembler edit routine interacts with CICS

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 CICS supplied macros DFHEIENT and DFHEIRET.



- Register 13 contains the address of a standard CICS working storage area as described by CICS supplied macro DFHEISTG.

An Assembler DSECT for DXEECS is shipped with QMF as DXEECSA, located in library QMF810.SDSQSAPE. Include this DSECT into your program using the Assembler COPY statement.

Return control to QMF by using the standard CICS RETURN command.

Translating 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).

Return control to QMF by using the CICS RETURN command; for example, EXEC CICS RETURN.

Assembling your program

During assembly, QMF edit exit interface control block DXEECSA, located in QMF sample library QMF810.SDSQUSRE, and the CICS macro library must be available.

Link-editing your program

Create a new QMF edit exit module DSQUECIC by including your edit program DSQUXCTA with EXEC CICS interface control modules DFHEAI

and DFHEAI0, 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 DSQUECIC.

The module DSQUECIC must be executable in 31-bit addressing mode.

Example JCL statements for translating, assembling and link-editing for CICS on z/OS

The following are example statements for translating, assembling, and link-editing your job for CICS.

```
 //SAMPASM JOB ...
//* Add a parameter PROGLIB to procedure DFHEITAL
//*      PROGLIB=&PROGLIB,
//TRNCOMLK EXEC PROC=DFHEITAL,PROGLIB='QMF810.SDSQLOAD'
//TRN.SYSIN DD *
      .
      Your program or modified copy of QMF sample DSQUXCTA
      .
/*
//* Provide access to QMF Edit Macro DXEECSA
//ASM.SYSLIB DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//LKED.SYSIN  DD  *
    INCLUDE SYSLIB(DFHEAI)
    INCLUDE CICSLOAD(DFHEAI0)
    ORDER DFHEAI,DFHEAI0
    ENTRY DSQUECIC
    MODE AMODE(31) RMODE(ANY)
    NAME DSQUECIC(R)
/*
```

Example program DSQUXCTA

The IBM-supplied example edit program in Assembler, named DSQUXCTA, is located in QMF sample library QMF810.SDSQSAPE on z/OS. The example program is heavily 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 program library and change its name to DSQUECIC.

How an Assembler edit routine interacts with QMF

The interface control block between QMF and the user edit interface DSQUEDIT is DXEECS. It 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.

Please refer to the DXEECSA file provided by QMF as a sample Assembler version of the DXEECS control block. This file is located in library QMF810.SDSQSAPE on z/OS.

Writing an edit routine in PL/I without language environment (LE)

You can write an edit routine in PL/I without language environment for native z/OS or TSO.

Writing an edit routine for native z/OS, TSO, or ISPF without LE

The QMF edit exit interface for PL/I in TSO, ISPF, or native z/OS consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSP
- Control program, which is shipped with QMF as DSQUXIP
- Control program, which is shipped with QMF as DSQUPLI
- Your edit exit program, which is named DSQUXDT

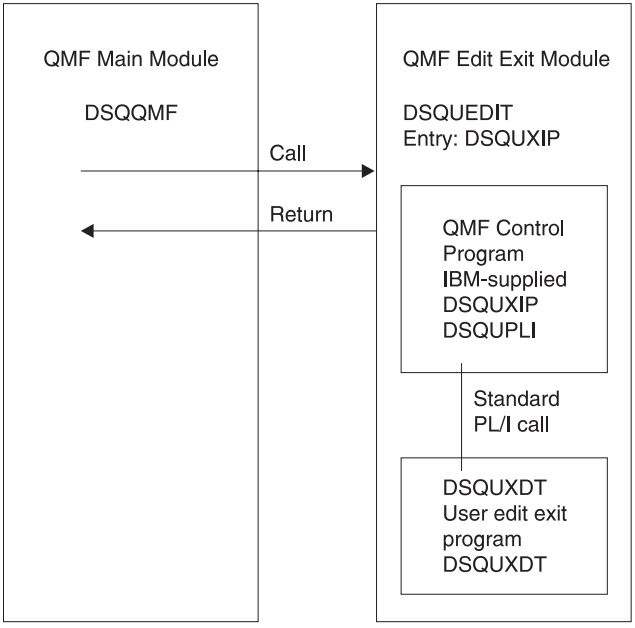


Figure 73. Program structure of a PL/I edit exit routine without LE

How a PL/I edit routine interacts with native z/OS, TSO, or ISPF

The user edit program is called as a PL/I external procedure using a standard PL/I CALL statement. The following parameters are provided in the indicated order:

1. DXEECS
2. Input data
3. Output data

An example procedure statement specifying parameters is as follows:

```
DSQUXDT:
    PROCEDURE(DXEECSF,ECSINPTF,ECSRSLTF) OPTIONS(REENTRANT);
```

A PL/I data structure is shipped with QMF as DXEECSP, located in library QMF810.SDSQSAPE. Include this data structure in your program.

Return control to QMF using a standard RETURN statement.

Compiling DSQUXDT and DSQUPLI

During the compile, QMF edit exit interface control block DXEECSP, located in QMF sample library QMF810.SDSQUSRE on z/OS must be available in a macro library.

Compile both programs with no STAE or SPIE macros. To do this, add the following statement to your PL/I program:

```
DCL PLIXOPT CHAR(15) VAR INIT('NOSTAE,NOSPIE') STATIC EXTERNAL;
```

Compile DSQUPLI with the MAIN option. Your edit exit program DSQUXDT must **not** specify MAIN.

Link-editing your program

Create a new QMF edit exit module DSQUEDIT by including your edit program DSQUXDT with the IBM-supplied control modules DSQUXIP and DSQUPLI, which are located in the QMF module library QMF810.SDSQLOAD. The module DSQUXIC must be specified as the entry point.

The 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.

We recommend 31-bit addressing mode.

Example statements for compiling and link-editing

The following are example statements for assembling and link-editing your job for TSO or z/OS.

```
//samPLI     JOB
//STEP1      EXEC IEL1CL
//* Provide Access to QMF Edit Macro DXEECSP
//PLI.SYSLIB   DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//PLI.SYSIN    DD  *
          .
          Your program or copy of QMF sample DSQUXDTP
          .
/*
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QM720.SDSQLOAD,DISP=SHR
//LKED.SYSIN    DD  *
          INCLUDE QMFLOAD(DSQUXIP)
          INCLUDE QMFLOAD(DSQUPLI)
```

```
          ENTRY DSQUXIP
          MODE  AMODE(31) RMODE(ANY)
          NAME  DSQUEDIT(R)
/*
```

Example program DSQUXDTP

The IBM-supplied example edit exit program in PL/I, named DSQUXDTP, is located in QMF sample library QMF810.SDSQSAPE. The example program is heavily commented; it can be browsed online, printed, or modified to meet your needs. If you plan to use this example program, copy it to your program library and change its name to DSQUXDT.

Writing an edit routine in PL/I with language environment (LE)

Use these instructions for writing an edit routine for native z/OS or TSO with language environment.

Writing an edit routine in PL/I for native z/OS, TSO, or ISPF with language environment (LE)

The QMF edit exit interface for PL/I in TSO, ISPF, or native z/OS with LE consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSP
- Control program, which is shipped with QMF as DSQUXILE
- Dynamic loaded LE preinitialization service program, which is named CEEPIPI
- Your edit exit program, which is named DSQUXDT

Figure 74 on page 245 shows the program structure of a PL/I edit exit routine in TSO, ISPF, or native z/OS.

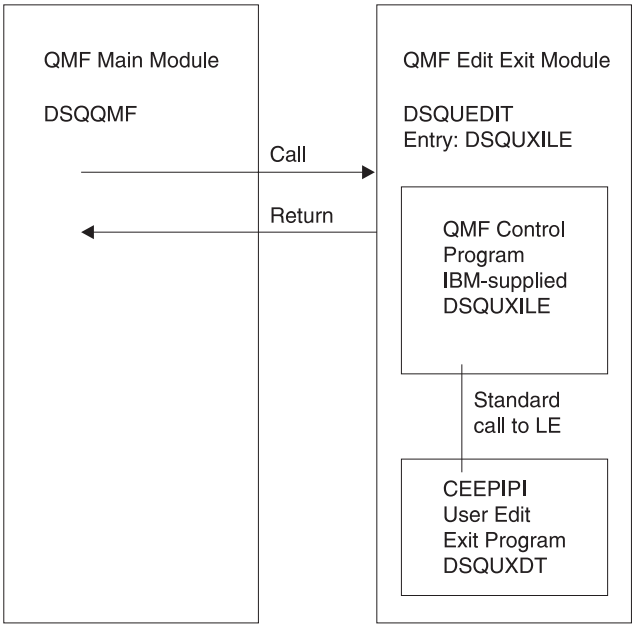


Figure 74. Program structure of a PL/I edit exit routine with LE

How a PL/I Edit routine interacts with native z/OS, TSO, or ISPF with LE

The user edit program is called as an LE subroutine. The following parameters are provided in the indicated order:

1. DXEECS
2. Input data
3. Output data

An example procedure statement specifying parameters is as follows:

```
DSQUXDT:
    PROCEDURE(DXEECSF,ECSINPTF,ECSRSLTF) OPTIONS(REENTRANT);
```

Compiling DSQUXDT

During the compile, QMF edit exit interface control block DXEECSP, located in QMF sample library QMF810.SDSQUSRE must be available in a macro library.

Compile the program with no STAE or SPIE macros. To do this, add the following statement to your PL/I program:

```
DCL PLIXOPT CHAR(15) VAR INIT('NOSTAE,NOSPIE') STATIC EXTERNAL;
```

Compile DSQUPLI with the MAIN option. Your edit exit program DSQUXDT must **not** specify MAIN.

Link-editing your program

Create a new QMF edit exit module DSQUEDIT by including your edit program DSQUXDT with the IBM-supplied control module DSQUXILE, located in the QMF module library QMF810.SDSQLOAD. The module DSQUXILE must be specified as the entry point.

The 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.

We recommend 31-bit addressing mode.

Example statements for compiling and link-editing

The following are example statements for assembling and link-editing your job for TSO or z/OS.

```
 //samPLI     JOB
 //STEP1      EXEC PLIXCL
 //* Provide Access to QMF Edit Macro DXEECSP
 //PLI.SYSLIB   DD  DSN=QMF810.SDSQUSRE,DISP=SHR
 //PLI.SYSIN    DD  *
           .
           Your program or copy of QMF sample DSQUXDTP
           .
/*
//* Provide Access to QMF & LE Interface Module
//LKED.QMFLOAD  DD DSN=QMF810.SDSQLOAD,DISP=SHR
//LKED.SYSLIB        DD DSN=SYS1.SCEELKED,DISP=SHR
 //LKED.SYSIN    DD  *
           INCLUDE QMFLOAD(DSQUXILE)
           ENTRY DSQUXILE
           MODE  AMODE(31) RMODE(ANY)
           NAME  DSQUEDIT(R)
 /*
```

Example program DSQUXDTP

The IBM-supplied example edit exit program in PL/I, named DSQUXDTP, is located in QMF sample library QMF810.SDSQSAPE. The example program is heavily commented; it can be browsed online, printed, or modified to meet your needs. If you plan to use the example program, copy it to your program library and change its name to DSQUXDT.

Writing an edit routine in PL/I for CICS on z/OS

The QMF edit exit interface for PL/I in CICS consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSP
- CICS command interface modules, which are shipped with CICS as DFHPL1OI

- Your edit exit program, which is named DSQUECIC

Figure 75 shows the program structure of a PL/I edit exit routine in CICS.

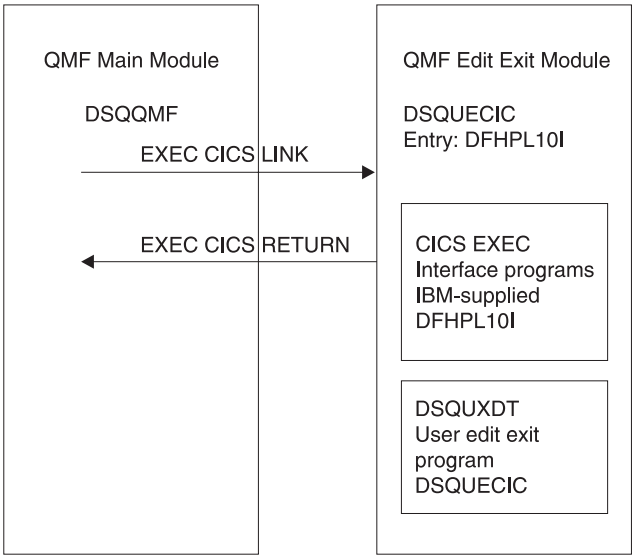


Figure 75. Program structure for PL/I edit exit routine in CICS

Example program DSQUXCTP

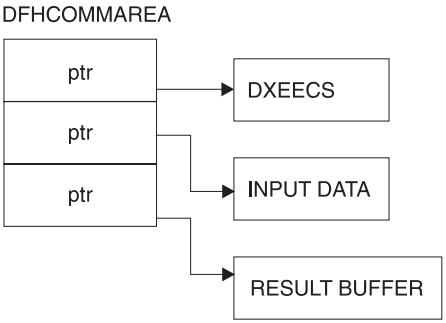
The IBM-supplied example edit program in Assembler, named DSQUXCTP, is located in QMF sample library QMF810.SDSQSAPE. The example program is heavily commented; you can print it, browse it online, or modify it to meet your needs. A PL/I data structure is shipped with QMF as DXEECSP, located in library QMF810.SDSQUSRE. Include this structure in your program.

How a PL/I edit routine interacts with CICS

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.

The CICS communications area DFHCOMMAREA is used to provide addresses to the user edit routine program parameters, DXEECS, input data, and output data as shown in the following diagram.

Figure 76. DFHCOMMAREA



After translation, the CICS translator provides a procedure statement that describes the CICS environment block DFHEIBLK. Provide a parameter that is a pointer to the CICS communications block DFHCOMMAREA such as the following example:

```
DSQUECIC:
    PROCEDURE(DFHCOMMP) OPTIONS(REENTRANT,MAIN);
```

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 PL/I program as follows:

```
/*****************************************************************/
  /*  CICS DFHCOMM ARE DESCRIPTION OF EDIT EXIT PARAMETERS          */
  /*****************************************************************/
  DECLARE
    DFHCOMMP PTR;
  DECLARE
    1  DFHCOMM BASED(DFHCOMMP),
       02 DFHCOMM_ECSPTR PTR,
       02 DFHCOMM_INPTR  PTR,
       02 DFHCOMM__OUTPTR PTR;
```

To provide addressability to the user edit routine control block DXEECS, input data area ECSINPT, and the result data area ECSRSLT, set the addresses of these data areas to the values located in DFHCOMMAREA as in the following example:

```
ECSPTR   = DFHCOMM_ECSPTR    /* ADDRESS OF DXEECS:
                                    EDIT CODE SPECIFICATIONS       */
    ECSINPTP = DFHCOMM_INPTR     /* ADDRESS OF INPUT DATA          */
    ECSRSLTP = DFHCOMM_OUTPTR    /* ADDRESS OF RESULT AREA         */
```

A PL/I data structure is shipped with QMF as DXEECSP, located in library QMF810.SDSQSAPE. Include this structure in your program.

Return control to QMF using a standard CICS RETURN command such as the following:

```
EXEC CICS RETURN;
```

Translating your program

Translate your program using the CICS translator for PL/I. During translation, CICS normally supplies an input parameter and data structure definition for the CICS environment control block EIB.

Compiling your program on z/OS

QMF edit exit interface control block DXEECSP, located in QMF sample library QMF810.SDSQUSRE, must be available in a macro library during the compile.

You must compile your program with no STAE or SPIE macros. To do this you should add the following statement to your PL/I program:

```
DCL PLIXOPT CHAR(15) VAR INIT('NOSTAE,NOSPIE') STATIC EXTERNAL;
```

Specify PL/I compiler option SYSTEM(CICS).

Link-editing your program

Create a new QMF edit exit module DSQUECIC by including the EXEC CICS interface control module DFHPL1OI, located in the CICS module library as distributed by the CICS product, and your edit exit program DSQUXCTP. Be sure to allocate PL/I libraries required for link-edit. Ensure DFHPL1OI or DFHPL1I is the first module in the edit exit module.

The module DSQUECIC must be executable in 31-bit addressing mode.

Example JCL statements for translating, compiling, and link-editing for CICS on z/OS

The following are example statements for translating, compiling, and link-editing your job for CICS.

```
//SAMPLI   JOB ...
//* Add a parameter PROGLIB to procedure DFHEITPL
//*      PROGLIB=&PROGLIB,
//TRNCOMLK EXEC PROC=DFHEITPL,PROGLIB='QMF810.SDSQLOAD'
//TRN.SYSIN DD *
      .
      Your program or modified copy of QMF sample DSQUXCTP
      .
/*
//* Provide access to QMF Edit Macro DXEECSP
//PLI.SYSLIB DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//LKED.SYSIN  DD  *
```

```
    REPLACE  PLISTART
    INCLUDE  CICSLOAD(DFHPL1OI)
    REPLACE  PLISTART
    ORDER    DFHPL1OI
    ENTRY    DFHPL1OI
    MODE     AMODE(31),RMODE(ANY)
    NAME     DSQUECIC(R)
/*
```

CICS program definition

When QMF is installed, the QMF edit exit program is installed with a program language of Assembler. To use the PL/I edit exit program, you must change the program language of module DSQUECIC to PL/I using the CICS program control table (PCT) macro or resource definition online (RDO).

Writing an edit routine in COBOL without language environment (LE)

You can write an edit routine in COBOL for native z/OS or TSO.

In this section, COBOL refers to VS COBOL II, COBOL/370, and COBOL for z/OS unless otherwise stated.

Writing an edit routine in COBOL for native z/OS, TSO, or ISPF without language environment (LE)

The QMF edit exit interface of COBOL consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSC
- Control program, which is shipped with QMF as DSQUXIC
- Your edit exit program, which is named DSQUXDT

Figure 77 on page 251 shows the program structure of a COBOL edit exit routine

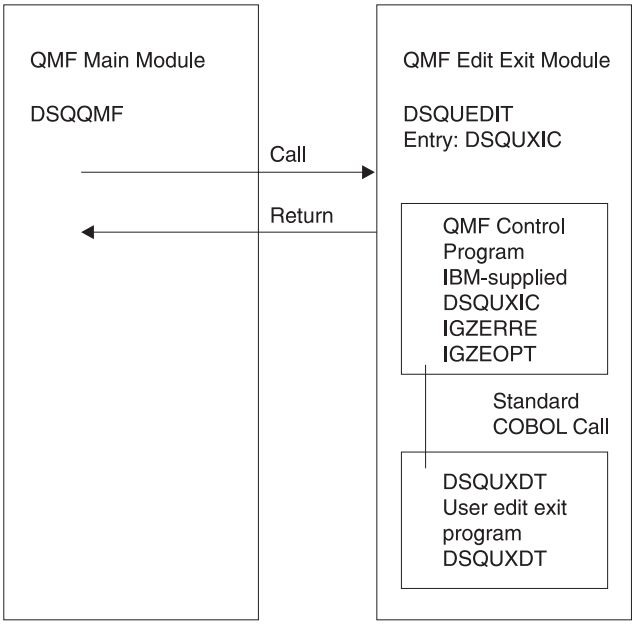


Figure 77. Program structure of a COBOL edit exit routine

Example program DSQUXDTC

The IBM-supplied example edit exit program in COBOL, named DSQUXDTC, is located in QMF sample library QMF810.SDSQSAPE on z/OS. 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 program library and change its name to DSQUXDT.

How a COBOL edit routine operates

The user edit program is called as a COBOL subprogram using a standard COBOL CALL statement. The following parameters are provided in the indicated order:

1. DXEECS
2. Input data
3. Output data

An example procedure statement specifying parameters is as follows:

```
PROCEDURE DIVISION
       USING DXEECS, ECSINPT, ECSRSLT.
```

Return control to QMF using standard subprogram GOBACK statement.

Compiling DSQUXDT

Compile DSQUXDT (the edit exit program you have written). During the compile, QMF edit exit interface control block DXEECSC, located in QMF sample library QMF810.SDSQUSRE on z/OS.

Select COBOL compiler options as follows:

COBOL II

Specify compiler options RENT, RES, NODYNAM, OBJECT, and LIB.

COBOL/370 or **IBM COBOL for z/OS**

Specify compiler options OBJECT, LIB, RENT, and NODYNAM.

QMF distributes the user edit routine control block DXEECSC using quotes as literal delimiters. You must use the QUOTE compiler option if you use the DXEECSC control block as distributed by IBM.

After compiling DSQUXDT, place the resulting load module in the QMF810.SDSQLOAD library.

Using the language environment run time library

When you use the Language Environment run time library with QMF user edit exit programs, consider the following:

- QMF does not require a new compile.
- LINK EDIT is required for any QMF user edit exit program that will be used with LE run time libraries.
- The QMF Assembler driver, DSQUXIC calls IGZERRE. See your IBM COBOL documentation for more information.

Assembling the run time options module

When you assemble the run time option macro IGZOPT, you must specify the COBOL run time option STAE=NO. (For the Language Environment options module, use TRAP=OFF in place of STAE=NO.) Include the resulting object module IGZEOPT in the QMF edit exit module DSQUEDIT.

Link-editing your program on z/OS

You create a new QMF edit exit module DSQUEDIT by including your edit exit program DSQUXDT with the IBM-supplied control module DSQUXIC, which is located in the QMF module library QMF810.SDSQLOAD. The module DSQUXIC must be specified as the entry point.

The 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.

Note: 31-bit addressing mode is recommended.

Example statements for compiling and link-editing on z/OS

The following are example statements for compiling and link-editing your job for TSO or native z/OS).

For COBOL II:

```
//samCOBOL JOB
//* Assemble run time option macro
//STEP1 EXEC PGM=IEV90,PARM='DECK,NOLOAD'
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&TEMPOBJ(IGZEOPT),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1,1)),DCB=(BLKSIZE=3120,LRECL=80,DSORG=PO)
//* Provide Access to Cobol run time option macro
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN DD *
      IGZOPT SYSTYPE=OS,STAE=NO
      END
//*
//STEP2      EXEC PROC=COB2UCL
//* Provide Access to QMF Edit Macro DXEECSC
//COB2.SYSLIB   DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//COB2.SYSIN    DD  *
          .
          Your program or copy of QMF sample DSQUXDTC
          .
/*
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QMF810.SDSQLOAD,DISP=SHR
//* Make sure COBOL library is concatenated after &&TEMPOBJ
//LKED.SYSLIB   DD DSN=&&TEMPOBJ,DISP=(OLD,PASS)
                 DD DSN=COB2LIB,DISP=(OLD,PASS)
//LKED.SYSIN    DD  *
          INCLUDE QMFLOAD(DSQUXIC)
          INCLUDE SYSLIB(IGZEOPT)
          INCLUDE SYSLIB(IGZERRE)
          ENTRY DSQUXIC
          MODE  AMODE(31) RMODE(ANY)
          NAME  DSQUEDIT(R)
/*
```

For COBOL/370 or IBM COBOL for z/OS:

```
//samCOBOL   JOB
//* Assemble run time option macro
//STEP1 EXEC PGM=IEV90,PARM='DECK,NOLOAD'
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&TEMPOBJ(IGZEOPT),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1,1)),DCB=(BLKSIZE=3120,LRECL=80,DSORG=PO)
//* Provide Access to Cobol run time option macro
```

```
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN DD *
      IGZOPT SYSTYPE=OS,STAE=NO
      END
//*
//STEP2      EXEC PROC=IGYWCL
//* Provide Access to QMF Edit Macro DXEECSC
//COBOL.SYSLIB  DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//COBOL.SYSIN   DD  *
          .
          Your program or copy of QMF sample DSQUXDTC
          .
/*
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QMF810.SDSQLOAD,DISP=SHR
//LKED.SYSLIB   DD ...
                 DD DSN=&&TEMPOBJ,DISP=(OLD,PASS)
//LKED.SYSIN    DD  *
          INCLUDE QMFLOAD(DSQUXIC)
          INCLUDE SYSLIB(IGZEOPT)
          INCLUDE SYSLIB(IGZERRE)
          ENTRY DSQUXIC
          MODE  AMODE(31) RMODE(ANY)
          NAME  DSQUEDIT(R)
/*
```

Writing an edit routine in COBOL with language environment (LE)

Use these instructions to write an edit routine in COBOL with language environment for native z/OS or TSO.

Writing an edit routine in COBOL for native z/OS, ISPF, and TSO with language environment (LE)

The QMF edit exit interface of COBOL in native z/OS and TSO consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSC
- Control program, which is shipped with QMF as DSQUXILE
- Your edit exit program, which is named DSQUXDT
- LE Preinitialization Service program, which is named CEEPIPI

Figure 78 on page 255 shows the program structure of a COBOL edit exit routine in native z/OS and TSO.

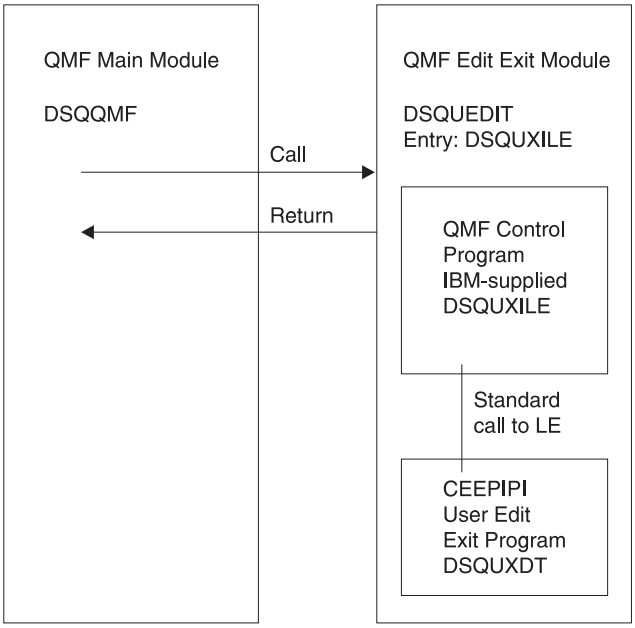


Figure 78. Program structure of a COBOL edit exit routine in TSO, ISPF, or native z/OS with LE

The edit control block DXEECSC and the sample COBOL program DSQUXCTC, as distributed by QMF, use quotes (″) to delimit literals. If your installation or program uses apostrophes (′) instead, you have to change DXEECSC or copy the structure to your program, changing quotes to apostrophes.

Example program DSQUXDTC

The IBM-supplied example edit exit program in COBOL, named DSQUXDTC, is located in QMF sample library QMF810.SDSQSAPE on and z/OS. 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 program library and change its name to DSQUXDT.

How a COBOL edit routine interacts with native z/OS, TSO, or ISPF in LE

The user edit program is called as an LE subroutine. The following parameters are provided in the indicated order:

1. DXEECS
2. Input data
3. Output data

An example procedure statement specifying parameters is as follows:

```
PROCEDURE DIVISION
       USING DXEECS, ECSINPT, ECSRSLT.
```

A COBOL data structure is shipped with QMF as DXEECSC, located in library QMF810.SDSQSAPE. Include this data structure in your program.

Return control to QMF using a standard subprogram GOBACK statement.

Compiling DSQUXDT

During the compile, QMF edit exit interface control block DXEECSC, located in QMF sample library QMF810.SDSQUSRE on z/OS, must be available in a macro library.

Compile the program with the following compile options:

`OBJECT, LIB, RENT, RES, and NODYNAM.`

Link-editing your program

Create a new QMF edit exit module DSQUEDIT by including your edit program DSQUXDT with the IBM-supplied control QMF module DSQUXILE (located in the QMF module library QMF810.SDSQLOAD on z/OS).

The module DSQUXILE must be specified as the entry point.

The 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.

Note: 31-bit addressing mode is recommended.

Example statements for compiling and link-editing on z/OS

The following are example statements for compiling and link-editing your job for TSO or native z/OS:

```
//samCOBOL   JOB
//STEP1     EXEC PROC=IGYWCL
//* Provide Access to QMF Edit Macro DXEECSC
//COBOL.SYSLIB  DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//COBOL.SYSIN   DD  *
```

Your program or copy of QMF sample DSQUXDTC:

```
/*
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QMF810.SDSQLOAD,DISP=SHR
//LKED.SYSLIB   DD ...
//                DD DSN=&&TEMPOBJ,DISP=(OLD,PASS)
//                DD DSN=SYS1.SCEELKED,DISP=SHR
//LKED.SYSIN    DD  *
          INCLUDE QMFLOAD(DSQUXILE)
```

```
            ENTRY DSQUXILE
            MODE  AMODE(31) RMODE(ANY)
            NAME  DSQUEDIT(R)
/*
```

Writing an edit routine in COBOL for CICS on z/OS

The edit exit interface for COBOL in CICS consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSC
- CICS command interface module, which is shipped with CICS as DFHECI
- Your edit exit program, which is named DSQUECIC

Figure 79 shows the structure of a COBOL edit exit routine in CICS.

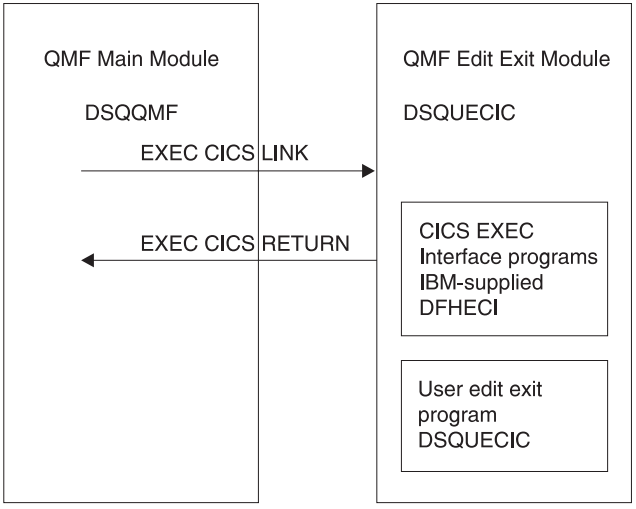
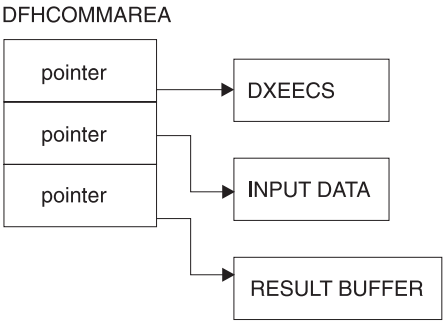


Figure 79. Program structure for a COBOL edit exit routine in CICS

How a COBOL edit routine interacts with CICS

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 COBOL. The CICS communications area DFHCOMMAREA is used to provide addresses to the user edit routine program parameters, DXEECS, input data, and output data as shown in the

following diagram.



After translation, the CICS translator provides a procedure statement that describes the CICS environment block DFHEIBLK and the CICS communications block, DFHCOMMAREA, like the following example:

```
PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA.
```

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 the 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.
```

A COBOL copy book is shipped with QMF as DXEECSC, located in library QMF810.SDSQSAPE on z/OS. Include this copy book in your program.

Return control to QMF using a standard CICS RETURN command such as the following:

```
EXEC CICS

          RETURN

     END-EXEC.
```

Translating 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 previous linkage section example.

Compiling

QMF edit exit interface control block DXEECSC, located in QMF sample library QMF810.SDSQUSRE, must be available in a macro library during the compile.

Specify COBOL compiler options RENT, RES, and NODYNAM, and run time options NOSTAE and NORTEREUS.

QMF distributes the user edit routine control block DXEECSC, using quotes as literal delimiters. You must use the QUOTE compiler option if you use the DXEECSC control block as distributed by IBM.

Link-editing

You create a new QMF edit exit module DSQUECIC by including your edit exit program DSQUXCTC with the EXEC CICS interface control module DFHECI, located in the CICS module library, as distributed by the CICS product. DFHECI must be the first module in the edit exit module and the entry point must be module DSQUECIC. Be sure to allocate COBOL libraries required for link-edit.

The module DSQUECIC must be executable in 31-bit addressing mode.

Example JCL statements for translating, compiling, and link-editing for CICS on z/OS

The following are example statements for translating, compiling, and link-editing your job for CICS.

```
//SAMCOBOL JOB ...
//* Add a parameter PROGLIB to procedure DFHEITVL
//*      PROGLIB=&PROGLIB,
//TRNCOMLK EXEC PROC=DFHEITVL,PROGLIB='QMF810.SDSQLOAD',
//      PARM.TRN='QUOTE',
//      PARM.COB='RENT,RES,NODYNAM,OBJECT,LIB,LIST,MAP,QUOTE'
//TRN.SYSIN DD *
      .
      Your program or modified copy of QMF sample DSQUXCTC
      .
```

```
/*
//* Provide access to QMF Edit Macro DXEECSC
//COB.SYSLIB DD  DSN=QMF810.SDSQUSRE,DISP=SHR
//LKED.SYSIN  DD  *
    INCLUDE SYSLIB(DFHECI)
    ORDER DFHECI
    ENTRY DSQUECIC
    MODE AMODE(31) RMODE(ANY)
    NAME DSQUECIC(R)
/*
```

CICS program definition on z/OS

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 DSQUECIC to COBOL using the CICS program control table (PCT) macro or resource definition online (RDO).

Example program DSQUCTC

The IBM-supplied example edit program in COBOL named DSQUXCTC is located in QMF sample library QMF810.SDSQSAPE on z/OS. The example program is heavily commented; it can be browsed online, printed, or modified to suit your needs.

How a COBOL edit routine interacts with QMF

The interface control block between QMF and the user edit interface DSQUEDIT is DXEECS. 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.

Handling double-byte character set data

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, read this section.

Among the characters represented by the Japanese DBCS are Latin characters and Katakana characters. A Latin character has these characteristics:

- The first (leftmost) 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.

Edit codes for DBCS data

You can use either Uxxxx or Vxxxx edit codes for DBCS data. The data that the edit routine receives is the same.

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 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 terminals, shift characters mark the start and end of a string of DBCS characters.

`So` denotes the shift character that introduces a DBCS string, and `Si` denotes the one that marks its end. `So` 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.

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 `So` and `Si` characters that bracket the DBCS strings. If there are no `So` or `Si` characters in ECSINPT, the string contains no DBCS data. For example, ECSINPT contains the following string:

```
ccccSodedededededededeSiccSodedededededeSi
```

Here, `c`, `d`, and `e` stand for any possible byte, and `So` 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 DBCS, 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.

Ensuring the edit routine returns the right results

Return the results in the ECSRSLT field, with trailing blanks for unused bytes. Make the results readable to the user's screen. 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 `So` and `Si` characters.

Overflowing the ECSRSLT field

Be careful not to overflow the ECSRSLT field, whose length is contained in the ECSRSLEN 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 rightmost 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 some terminals, the following rules apply:

- If the report is displayed on the screen, the `Si` and `So` characters embedded in a user's results also appear on the terminal.
- The `Si` and `So` characters appear either as blanks or as special symbols. There is one special symbol for `Si` and another for `So`.
- Blanks appear instead of the symbols unless the user presses a certain combination of keys.

For other devices, the rules can be slightly different.

Instructions for using DBCS characters in the online help say not to use certain DBCS characters in queries and QMF commands. The same restriction does not apply to the formatted data returned by an edit routine. Any legitimate DBCS character can be returned in the ECSRSLT field.

Chapter 19. Controlling QMF resources using a governor exit routine

Note: This chapter contains General Use Programming Interface and Associated Guidance Information.

A governor exit routine helps you limit end-user activity and control use of computer resources at your installation. IBM supplies a governor exit routine for QMF with default limits. For example, you can limit he number of rows a user can retrieve from the database. You can use this default exit routine, modify the routine, or write one of your own using Assembler.

Using a governor exit routine on z/OS

On z/OS, default limits are provided for the amount of time spent running a QMF command.

You can use the DB2 governor with the QMF governor to monitor the processor time used when dynamically running SELECT, INSERT, UPDATE, and DELETE queries. You can also use the DB2 governor independently.

You can also use the QMF High Performance Option/Manager (HPO/Manager) to manage and control QMF session activity. With HPO/Manager you also have a real-time user interface to QMF session activity and a query analyzer that estimates a query's resource use before it is run. The HPO/Manager overrides the QMF governor. For more information about the HPO feature, see *DB2 QMF High Performance Option User's Guide for TSO/CICS*.

Using the IBM-supplied governor exit routine

The governor exit routine supplied for CICS (DSQUEGV3) controls how many rows a user can retrieve from the database. The governor exit routine supplied for TSO, ISPF, and native z/OS (DSQUEGV1) controls how many rows a user can retrieve from the database or the processor time used running a QMF command. The governor exit routine is shipped with two predefined values for the number of rows:

- A row prompt value warns users when the number of rows retrieved reaches 25,000, at which time the user sees the message shown below.

```
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. The n symbol in the figure represents an NLID from Table 1 on page ix

Important: Database activity is not suspended when a cancellation prompt is displayed. DB2 continues to fetch rows and use processor time.

- A row limit value 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. When the IBM-supplied governor cancels data retrieval, the user sees the message shown in below.

```
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. For example, if your procedure contains a command that requires the report to complete (such as ERASE), you receive the message shown below.

```
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.

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 value warns users when the processor time for the cycle reaches six minutes, at which time the user sees the message shown below.

```
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 (z/OS). The n symbol in the figure represents an NLID from Table 1 on page ix

- A time limit value cancels the command when 24 minutes of processor time are used during the cycle.

Activating the default limits

Follow this procedure to 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 the QMF activity when the number of rows retrieved reaches 100 ,000:

1. Run the query shown in Figure 84 from the SQL query panel.

```
UPDATE Q.RESOURCE_VIEW
SET INTVAL=0
WHERE RESOURCE_OPTION='SCOPE' AND
      RESOURCE_GROUP='SYSTEM'
```

Figure 84. Activating default values for the IBM-supplied governor

2. Set a value of SYSTEM for the RESOURCE__GROUP field of the user's profile. For example, the UPDATE statements in Figure 85 activate default values for user JONES (using English QMF) and user SCHMIDT (using German QMF).

   Important: Always specify a value for the TRANSLATION column, or you might change more rows in Q.PROFILES than you intend.

```
Base QMF (English)
        German NLF
UPDATE Q.PROFILES
        UPDATE Q.PROFILES
SET RESOURCE__GROUP = 'SYSTEM'
        SET RESOURCE__GROUP = 'SYSTEM'
WHERE CREATOR='JONES' AND
        WHERE CREATOR='SCHMIDT' AND
TRANSLATION='ENGLISH'
        TRANSLATION='DEUTSCH'
```

Figure 85. Updating a user's resource group

   Important: If you start QMF with a DSQSPRID parameter value of TSOID, the resource group name is the user ID.

3. Instruct users to reconnect to the database to activate the new values. This can be done with a DB2 CONNECT command, or they can end their current QMF session and begin another to activate the new resource group.

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 266. Then see the procedure in "Defining your own resource limits" on page 269.

How a governor exit routine controls resources

The governor uses two types of information to control resources.

- Information about the resource limits you set for a user, defined in a resource control 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 IBM-supplied control blocks DXEGOVA and DXEXCBA.

How the governor knows what the resource limits are: Each row of the IBM-supplied Q.RESOURCE_TABLE contains:

- 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`) you want to limit for the group of users named in RESOURCE_GROUP.
- Values (`INTVAL`, `FLOATVAL`, or `CHARVAL`) that define the limit for the resource option. Resource options can have integer values, floating-point values, or character values.

Table 51 shows the structure of the Q.RESOURCE_TABLE as it is shipped by IBM. Q.RESOURCE_TABLE has the index Q.RESOURCE_INDEX. Keyed columns are RESOURCE_GROUP and RESOURCE_OPTION.

If you are migrating from an older QMF release: The older QMF releases do not include Q.RESOURCE_INDEX.

The Q.RESOURCE_TABLE is shipped by IBM with a predefined resource group called SYSTEM. The SYSTEM resource group has three predefined resource options for CICS. The group has additional time options for TSO, ISPF, or native z/OS batch. Use the CHARVAL column to indicate the limits defined in each row, as shown.

Table 51. Default resource group and options for the IBM-supplied governor exit common to all

GROUP	OPTION	INTVAL	FLOATVAL	CHARVAL
SYSTEM	SCOPE	-	-	Indicate whether governor is active
SYSTEM	ROWLIMIT	100,000	-	Cancel after fetching 100,000 rows
SYSTEM	ROWPROMPT	25,000	-	Prompt user after fetching 25,000 rows

Table 52. Options for the IBM-supplied governor exit for TSO, ISPF, or native z/OS batch

GROUP	OPTION	INTVAL	FLOATVAL	CHARVAL
SYSTEM	TIMELIMIT	1440	-	Cancel after 24 minutes CPU
SYSTEM	TIMEPROMPT	360	-	Prompt user after 6 minutes CPU
SYSTEM	TIMECHECK	900	-	15 minute interval between time check

SCOPE = 0
Activates governing for a particular resource group.

Any non-zero value for SCOPE, including a null, deactivates governing for the resource group.

ROWLIMIT = 100,000
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.

ROWPROMPT = 25,000
Warns the user when 25, 000 database rows have been retrieved.

The three additional options provided in TSO, and native z/OS batch are:

TIMELIMIT = 1440
If the user decides to continue when warned, the governor exit routine cancels the command after 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.

Processor time: Processor time refers to the jobstep time plus the SBR (Service Request Block) time.

TIMEPROMPT = 360
Warns the user when 6 minutes of processor time have elapsed. Evaluated after a TIMECHECK interval is processed.

TIMECHECK = 900
Specifies 15 minutes of real time between time checks or prompting or canceling.

IBM also supplies a view of this table, called Q.RESOURCE_VIEW, that 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. The governor exit routine uses this resource information to help determine when the user reaches a resource limit.

How the governor knows when you reach a resource limit: On a call to the governor exit routine, QMF queries Q.RESOURCE_VIEW, which shows what resource limits are defined in the resource control table for the resource group to which the user belongs. To determine the resource group, QMF checks the value of the RESOURCE_GROUP field of the user's row in the Q.PROFILES table and checks Q.RESOURCE_VIEW for a matching value.

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.

Figure 86 shows how the governor limits use of resources.

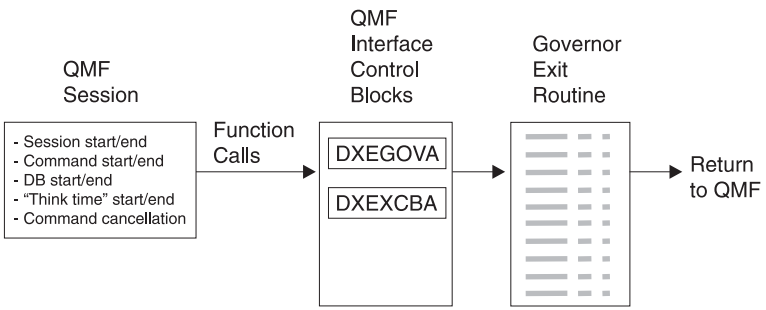


Figure 86. How a governor exit routine works with QMF for TSO/CICS

QMF calls the governor exit routine at a number of different points within the QMF session. These calls are called function calls. For more information about function calls, see "Points at which QMF calls the governor" on page 277.

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 IBM-supplied 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 IBM-supplied governor exit routine also displays a warning before canceling the activity, as shown in

Figure 81 on page 264. See "Defining your own resource limits" for how to activate this warning if you are not using the default values for the number of rows retrieved.

The IBM-supplied 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.

Defining your own resource limits

This section explains how to create a new resource group, for which the resource is the number of rows retrieved from the database. If you want to define resource limits other than the number of rows, you need to modify the IBM-supplied governor exit routine or write an exit routine of your own. See "Modifying the IBM-supplied governor exit routine or writing your own" on page 273 for more information on the facilities you can use.

Use the following procedure to add a resource group to the resource control table. This procedure adds a resource group named GROUP1, where the governor prompts a user in GROUP1 when the number of rows reaches 10, 000, and cancels the user's activity when the number of rows reaches 10,000; it cancels the user's activity when the number of rows reaches 15 ,000. For TSO 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.

1. Run the query in Figure 87 to 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

```
INSERT INTO Q.RESOURCE__VIEW (RESOURCE__GROUP,RESOURCE__OPTION,INTVAL)
    VALUES('GROUP1','ROWPROMPT',10000)
```

Figure 87. Activating prompting for row limit

2. Run the query in Figure 88 to set the number of rows at which the governor cancels the user's activity.

```
INSERT INTO Q.RESOURCE__VIEW (RESOURCE__GROUP,RESOURCE__OPTION,INTVAL)
    VALUES('GROUP1','ROWLIMIT',15000)
```

Figure 88. Activating cancellation of activities when user reaches row limit

3. Run the query in Figure 89 to set the processor time that elapses before 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 time elapsed, skip to step 4.

```
INSERT INTO Q.RESOURCE__VIEW (RESOURCE__GROUP,RESOURCE__OPTION,INTVAL)
    VALUES('GROUP1','TIMEPROMPT',300)
```

Figure 89. Activating prompting for time limit

4. Run the query in Figure 90 to set the 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 90. For TSO and native OS/390 and z/OS batch: Activating cancellation of activities when user reaches time limit

5. Run the query in Figure 91 to 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 91. For TSO and native z/OS batch: Activating time interval check

6. Run the query shown in Figure 92 to turn on governing for the GROUP1 resource group. SCOPE is a resource option that activates or deactivates governing. Each resource group in the 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 92. Turning on the governor for a particular resource group

7. Run a query similar to the one in Figure 93 on page 271 to 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 93. Updating a user's resource group

If you are using an NLF: Use a similar query to update a user's profile in an NLF environment, but use a TRANSLATION value from Table 1 on page ix.

8. Instruct the user whose profile you updated to end the current QMF session and start another to activate the new values. This can be done with a DB2 CONNECT command or they can end their current QMF session and begin another to activate the new values.

Creating your own resource control table

You can create your own table or rename the Q.RESOURCE_TABLE. You can also include additional columns in the table you create, if Q.RESOURCE_VIEW is the view defined in this table, and if the table includes all of the columns shown inTable 53 on page 272.

Figure 94 shows an example of SQL statements you might use to create a table called MY_RESOURCES. Substitute your own table, column, and table space names in the query. Before creating a new table, ensure you erase the Q.RESOURCE_TABLE from the database, because Q.RESOURCE_VIEW is defined in this table:

```
DROP TABLE Q.RESOURCE_TABLE
```

Dropping the Q.RESOURCE__TABLE also drops Q.RESOURCE_VIEW from the database, so you need to recreate both the table and the view, as shown in Figure 94 and Figure 95 on page 272. Under TSO, substitute your own table space name for SPACE1.

```
CREATE TABLE MY_RESOURCES
  (GROUP_NAME    CHAR(16) NOT NULL,
   CONSTRAINT    CHAR(16) NOT NULL,
   INTEGER       INTEGER,
   FLOAT_VALUE   FLOAT,
   CHARACTER     VARCHAR(80))
IN TBSPACE1
```

Figure 94. Creating a resource control table or renaming Q.RESOURCE_TABLE

When running QMF for TSO/CICS, you automatically invalidate the QMF application plan when you drop the view. For this reason, you should work

outside QMF when you drop and recreate the resource table and view. Choose a time when QMF is inactive, and use DB2's DB2I facility. DB2I lets you carry out the work interactively.

If you do not use the IBM-supplied table space, you must create your own. If you rebind the QMF authorization plan explicitly, you also need the BIND privilege on the plan. You can find information on the needed authority for each of your SQL commands in the *DB2 UDB for z/OS SQL Reference*.

Always recreate 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.

Figure 95 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 figure.

```
CREATE VIEW Q.RESOURCE_VIEW
 (RESOURCE_GROUP, RESOURCE_OPTION, INTVAL, FLOATVAL, CHARVAL)
  AS SELECT GROUPNAME, CONSTRAINT, INTEGER, FLOAT_VALUE, CHARACTER
  FROM MY_RESOURCES
```

Figure 95. Redefining the Q.RESOURCE_VIEW

After you create the view, you must grant the SELECT privilege on Q.RESOURCE_VIEW to PUBLIC. Then test the new view; you can test the view using SPUFI. Finally, rebind the QMF authorization plan.

Table 53. Structure of the Q.RESOURCE_TABLE table

Column name	Data type	Length (bytes)	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.
RESOURCE__OPTION	CHAR	16	No	Your own name for a resource you want to monitor.
INTVAL	INTEGER		Yes	Reflects resource limit for resource options that have integer values. For example, number of rows retrieved from the database is a resource that has an integer value.

Table 53. Structure of the Q.RESOURCE_TABLE table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
FLOATVAL	FLOAT		Yes	Reflects resource limit for resource options that have floating point values. FLOATVAL is null for the IBM-supplied governor.
CHARVAL	VARCHAR	80	Yes	Reflects 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 IBM-supplied governor.

Modifying the IBM-supplied governor exit routine or writing your own

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 IBM-supplied governor exit routine or write your own by doing the following:

1. 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 276 explains this step.
2. Pass resource control information to the governor exit routine and store this information. "Passing resource control information to the governor exit" on page 284 explains this step.
3. Establish addressability to the QMF cancellation service to cancel activities. "Canceling user activity" on page 297 explains this step.
4. Establish addressability to the QMF message service to provide messages for activities that have been canceled. "Providing messages for canceled activities" on page 298 explains this step.

Modifying the governor exit on z/OS

For TSO, and native z/OS batch, assemble, and link-edit your governor exit routine, whether you modified the IBM-supplied governor exit routine or wrote your own.

For CICS, translate, assemble, and link-edit your governor exit routine, whether you modified the IBM-supplied governor exit routine or wrote your own. "Assembling, translating, and link-editing your governor exit routine in CICS on z/OS" on page 301 explains this step.

Program components of the governor exit routine

Before you begin modifying or writing your own governor exit routine, you need to know the names of the governor exit routine components and what purpose each component serves.

Table 54 shows these components, whose names vary according to which language you installed (English or an NLF). Replace the *n* symbol in the following names with the NLID (from Table 1 on page ix) that matches the NLF you're using. In the component names, a 1 represents TSO and native z/OS batch.

Table 54. IBM-supplied governor components

Member Name	Library	Function
	TSO, ISPF, and native z/OS	
DSQUnGV1	QMF810.SDSQLOAD	Load module for TSO, and native z/OS batch
DSQUnGV1	QMF810.SDSQUSRn	Source code for governor exit routine for TSO, and native z/OS batch
DXEUnGV1	QMF810.SDSQUSRn	Contains text and related definitions for the governor prompts and cancellation messages in TSO, and native z/OS batch
	CICS on z/OS	
DSQUnGV3	QMF810.SDSQLOAD	Load module for CICS
DSQUnGV3	QMF810.SDSQUSRn	Source code for governor exit routine for CICS
DXEUnGV3	QMF810.SDSQUSRn	Contains text and related definitions for the governor cancellation message in CICS
DXEUnGM	QMF810.SDSQUSRn	Contains BMS map for the governor prompts in CICS.
DXEGOVA	QMF810.SDSQUSRn	DSECT for the DXEGOVA control block
DXEXCBA	QMF810.SDSQUSRn	DSECT for the DXEXCBA control block.

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 DSQU*n*GVx for each language environment. For example, if you have both English and German installed, use the module DSQUEGV1 for English in TSO, and native z/OS batch and the module DSQUDGV1 for German in TSO, 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.

How TSO, and native z/OS interact with the governor exit routine

At the start of a user's session, QMF issues a LOAD command to bring the governor into the user's virtual storage. For performance reasons, an Assembly 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.

Throughout this chapter, the load module library QMF810.SDSQLOAD is assumed to be in a library concatenated to the user's STEPLIB data set.

Figure 96 shows the program structure of a governor exit routine.

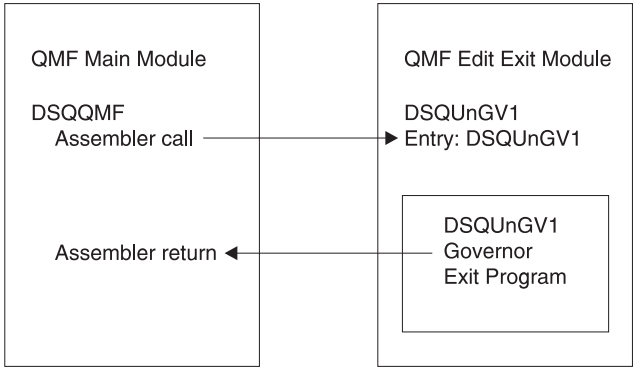


Figure 96. TSO or native z/OS processing that interacts QMF with the governor exit

How CICS interacts with the governor exit routine

At the start of a user's session, QMF issues an EXEC CICS LOAD command to bring the governor into the user's virtual storage. 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. Assembling and link-editing this module are discussed in "Assembling, translating, and link-editing your governor exit routine in CICS on z/OS" on page 301.

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

- CICS-supplied 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 DSQU*n*GV3

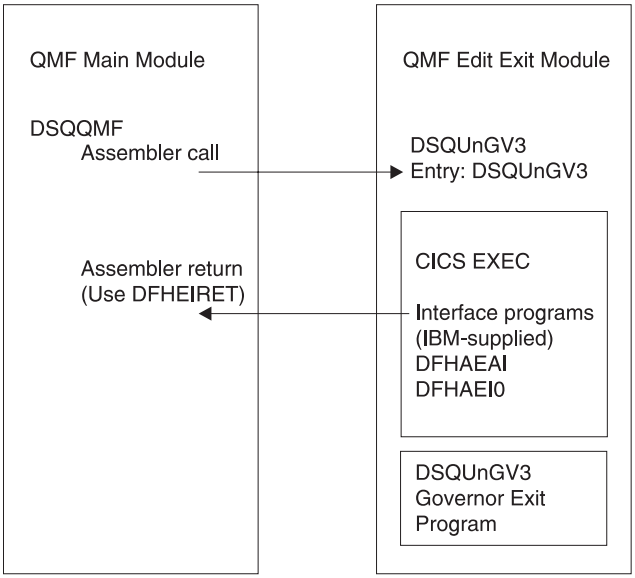


Figure 97. CICS processing that interacts QMF with the governor exit

The governor exit routine executes on the same program level as the main QMF program.

The entry point to the governor exit routine is DSQU*n*GV3. When it calls the governor exit routine, QMF always branches to the address returned by CICS as the result of an EXEC CICS LOAD command.

If the load fails or the module does not support 31-bit addressing mode, QMF issues a warning message, disables the governor exit, and continues the session without the governor. Assembling and link-editing this module are discussed in "Assembling, translating, and link-editing your governor exit routine in CICS on z/OS" on page 301.

How and when QMF calls the governor exit routine

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.

z/OS

Follow these instructions for z/OS.

Points at which QMF calls the governor

Function calls to the governor exit routine either precede or follow a specific type of QMF activity. For example, QMF passes control to the governor exit before and after running a command.

When it calls the governor, QMF always branches to an entry point named DSQU*n*GVx. Therefore, you cannot use the entry point to determine the type of exit. Use instead the control-block field GOVFUNCT. Its value is a positive integer that identifies the type of exit.

- **At the beginning and end of a QMF session**

  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 QMF's query of Q.RESOURCE_VIEW.

- **After a new connection is made to the database**

  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.

  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.

- **Before and after running a command**

  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.

- **Before database activity starts and when it ends**

  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.

  When QMF retrieves data, it fits the maximum number of rows possible into a buffer that has a minimum size of 4K. 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.

QMF also calls the governor when SQL, QBE, or prompted queries are submitted using RUN QUERY, or when QMF is running queries started by a command. For example, a SAVE DATA command might result in DELETE, CREATE, and INSERT queries. 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. See "Solving performance problems" on page 335 for more information on handling problems associated with completing the data object.

The following QMF commands always force database activity:

- DISPLAY table commands
- The EDIT TABLE command for the Table Editor
- The ERASE command for a table
- The EXPORT TABLE command
- The IMPORT command to a table
- The PRINT command for a table or view
- The RUN command for queries
- The SAVE DATA command (which forces an implicit CREATE TABLE query)
- Scrolling commands that result in fetching data when a report is being displayed
- Data retrieval operations (fetch operations)

- **Before and after the user makes a choice**

  At various points in a session, QMF waits for users to make decisions. The time QMF spends waiting is known as think time.

  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.

  Any of the following activities lead to think time:

  - Displaying a QMF panel between running commands
  - Displaying help panels
  - Displaying confirmation prompt panels; for example, when the user is about to erase something by issuing the SAVE command that replaces the object
  - Displaying command prompt panels; for example, when the user enters `DISPLAY ?`
  - Displaying the LIST prompt panel
  - Displaying ICU and EXTRACT panels
  - Running the EDIT PROC and EDIT QUERY functions

- **At initiation of an abnormal ending**

  QMF calls the governor just before it initiates an abnormal ending. The governor can perform the cleanup necessary before the abend processing begins. The actions might be similar to those during the session end.

For the IBM-supplied governor exit routine, QMF uses the GOVFUNCT field of the DXEGOVA control block to pass information about the type of function call. Each type of function call has a specific value for the GOVFUNCT field. These values are shown in Figure 98.

What happens upon entry to the governor exit routine

QMF calls the governor exit routine by branching to the address of the entry point DSQU*n*GV1 (TSO), or DSQU*n*GV3 (CICS).

Branching to the CICS entry point DSQUnGV3: 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 CICS-supplied macros DFHEIENT and DFHEIRET Figure 98 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.

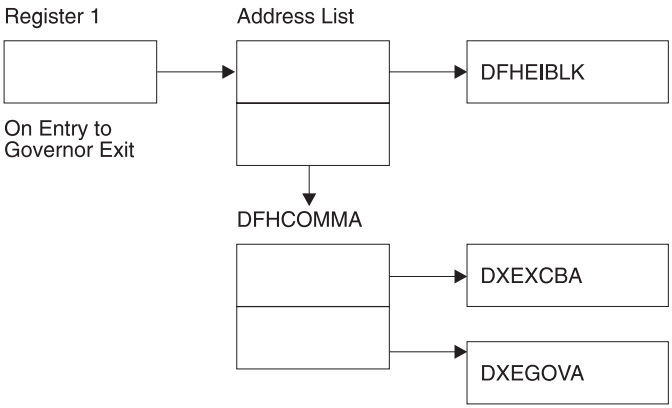


Figure 98. Contents of Register 1 on a call to the governor exit routine

- 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 using any EXEC CICS commands that change the environment

(for example, CICS HANDLE CONDITION). If you need to use the CICS HANDLE CONDITION, use EXEC CICS PUSH and EXEC CICS POP to save and restore them.

Begin the governor program with code similar to that shown below.

```
DSQUEGV3 TITLE 'QMF GOVERNOR EXIT ROUTINE'
DFHEISTG DSECT
DSQUEGV3 DFHEIENT CODEREG=(12),DATAREG=(13),EIBREG=(10)
         B     FDENTRY              BRANCH AROUND CONSTANTS
*
MODNAME  DC    C'DSQUEGV3'          MODULE NAME
         DC    C' '
         DC    C'&SYSDATE '         DATE OF ASSEMBLY
         DC    C'&SYSTIME '         TIME OF ASSEMBLY
         DS    0H
*
FDENTRY  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 99. Sample code at the start of a governor (for CICS)

The code in Figure 99 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, DXEXCB, DXEGOV, 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 IBM-supplied governor.

When processing is complete, the governor returns control to QMF using the standard CICS return as specified by the CICS macro DFHEIRET.

Attention: Do not use the command EXEC CICS RETURN. This ends the QMF session without releasing QMF resources.

The governor program ends with code similar to Figure 100.

```
:
*
         XR    R15,R15           ZERO RETURN CODE
         DFHEIRET RCREG=15
*
```

Figure 100. Endcode for the governor program

Branching to the entry point: QMF calls the governor exit routine by branching to the address of the entry point DSQUEGV1 (TSO). 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 control block DXEXCBA; the other for the control block DXEGOVA.

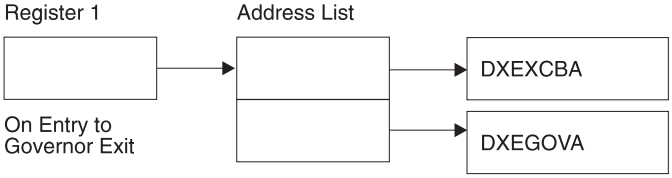


Figure 101. Contents of Register 1 on a call to the governor exit routine

- 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 that shown in Figure 102 on page 282. The code sample is from the IBM-supplied governor for TSO or native z/OS.

```
DSQUEGV1 CSECT
         USING *,R15
         B     FDENTRY              BRANCH AROUND CONSTANTS
         DC    C'DSQUEGV1'          MODULE NAME
         DC    C' '
         DC    C'&SYSDATE '         DATE OF ASSEMBLY
         DC    C'&SYSTIME '         TIME OF ASSEMBLY
         DS    0H
*
FDENTRY  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 102. Sample code at the start of a governor (for TSO, ISPF, or native z/OS)

The code in Figure 102 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, DXEXCB, DXEGOV, 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 IBM-supplied governor.

After processing a call, the governor returns control to QMF in the standard way; that is, you must use the standard epilog and prolog. In the IBM-supplied governor, the following code does this:

```
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
```

Establishing addressability for function calls

Because QMF always branches to an entry point named DSQU*n*GV1 (TSO), or DSQU*n*GV3 (CICS) when it calls the governor, you cannot use these entry points to determine the type of function call; instead, use the GOVFUNCT field of the DXEGOVA control block.

In the IBM-supplied governor exit routine, GOVFUNCT 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 GOVFUNCT 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 Figure 103. 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 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
```

Figure 103. Character and numeric values for the GOVFUNCT field of DXEGOVA

To improve performance in your own exit routine, you can follow the convention the IBM-supplied governor uses, and equate the values of GOVFUNCT 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.

Figure 104 on page 284 shows an example of some code that identifies branch addresses for the IBM-supplied 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 104. Identifying the type of function call and branching to the appropriate address

Passing resource control information to the governor exit

If you have not done so already, read the following section, which describes how to set up resource control information in a format the governor can use:"Defining your own resource limits" on page 269.

QMF passes resource control information using two control blocks named DXEGOVA and DXEXCBA. Their addresses are passed to the governor on every function call. The DSECT DXEXCBA (shipped as DXEXCBA) and the DSECT DXEGOVA (shipped as DXEGOVA) are located in the SDSQUSRE MACLIB. Include these DSECTs in your program using the assembler COPY statement.

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.

Table 55 on page 285 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 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 QMF in the operating system you are running in.

Table 55. Fields of the DXEGOVA interface control block to the governor

Field	Data type	Purpose
GOVCADDR	A	Contains the address to branch to for canceling an activity.
GOVFUNCT	XL1	Indicates the type of function call. Possible values are: • GOVINIT (session initialization); GOVTERM (session termination) • GOVSCMD (start command); GOVECMD (end command) • GOVCONN (connect command) • GOVSDBAS (start database retrieval operation); GOVEDBAS (end database retrieval operation) • GOVSACTV (suspend QMF activity for user think time); GOVRACTV (resume QMF activity) • GOVABEND (start of an abnormal ending)
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.
GOVRESC	10XL128	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. • If the resource group has less than 10 rows, unused blocks are those at the end of the field. • If the resource group has more than 10 rows, use the field named GOVNEXTR (in the GOVRESCT DSECT) to access additional rows.

Table 55. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data type	Purpose
GOVRESCT	DSECT	Describes the block of storage containing information on one of the user's rows of the resource control table. **GOVOPTN(CL16)** 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. **GOVNULLI(H)** Null indicator for INTVAL column. **GOVINTVL(F)** Value of INTVAL column. **GOVNULLF(H)** Null indicator for FLOATVAL column. **GOVFLOAT(D)** Value of FLOATVAL column. **GOVNULLC(H)** Null indicator for CHARVAL column. **GOVCHLEN(H)** Length of data in CHARVAL column. **GOVCHAR(CL80)** Value in CHARVAL column. **GOVNEXTR(A)** Points to the block of data for the next resource table row. Contains zero if this is the last row. 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.
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.

```
*************************************************************************** 00001000
*                                                                         * 00002000
*        CONTROL BLOCK NAME: DXEGOVA                                      * 00003000
*                                                                         * 00004000
*        FUNCTION:                                                        * 00005000
*                                                                         * 00006000
*            THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND           * 00007000
*            THE GOVERNOR EXIT ROUTINE.                                   * 00008000
*                                                                         * 00009000
*        STATUS: VERSION 8 RELEASE 1 LEVEL 0                              * 00010000
*                                                                         * 00011000
*        INNER CONTROL BLOCKS: NONE                                       * 00012000
*                                                                         * 00013000
*        CHANGE ACTIVITY: NA                                              * 00014000
*                                                                         * 00015000
*        CHANGE DATE: NA                                                  * 00016000
*                                                                         * 00017000
*************************************************************************** 00018000
*                                                                         00019000
DXEGOVA  DSECT                                                           00020000
         DS    0D                                                        00021000
GOVNAME  DS    CL8           -- CONTROL BLOCK IDENTIFICATION             00022000
         SPACE                                                           00023000
GOVEXCTL DS    XL72          -- EXIT CONTROL                             00024000
         ORG   GOVEXCTL                                                  00025000
GOVFUNCT DS    XL1           ----- FUNCTION CODE                         00026000
GOVINIT  EQU   1             -------- INITIALIZATION OF SESSION          00027000
GOVTERM  EQU   2             -------- TERMINATION OF SESSION            00028000
GOVSCMD  EQU   3             -------- START COMMAND                      00029000
GOVECMD  EQU   4             -------- END COMMAND                        00030000
GOVCONN  EQU   5             -------- CONNECT COMMAND                    00031000
GOVSDBAS EQU   6             -------- START DATA BASE                    00032000
GOVEDBAS EQU   7             -------- END DATA BASE                      00033000
GOVSACTV EQU   8             -------- SUSPEND QMF ACTIVITY              00034000
GOVRACTV EQU   9             -------- RESUME QMF ACTIVITY              00035000
GOVABEND EQU   10            -------- QMF ABEND OPERATION               00036000
GOVPAD10 DS    CL7           ----- RESERVED FIELD                        00037000
         SPACE                                                           00038000
GOVCADDR DS    A             ---- ADDR TO BRANCH TO FOR CANCELLATION     00039000
         SPACE                                                           00040000
GOVOROWS DS    F             ---- NUMBER OF OPTION ROWS RETRIEVED        00041000
         SPACE                                                           00042000
GOVSQLRC DS    F             ---- RESOURCE TABLE SQL RETURN CODE         00043000
         SPACE                                                           00044000
GOVSQLCA DS    A             ---- ADDRESS OF SQLCA FOR ERROR CONDITION 00045000
         SPACE                                                           00046000
GOVGROUP DS    CL16          ---- GROUP NAME                             00047000
GOVPAD20 DS    CL32          ---- RESERVED FIELD                         00048000
```

Figure 105. The DXEGOVA control block (Part 1 of 2)

```
SPACE                                                                  00049000
GOVUCTL  DS    XL304         -- USER CONTROL AREA                      00050000
         ORG   GOVUCTL                                                 00051000
GOVUSERS DS    CL2048        ----- USER SCRATCH PAD AREA               00052000
GOVPAD30 DS    CL48          ----- RESERVED FIELD                      00053000
         SPACE                                                         00054000
         DS    0D                                                      00055000
GOVRESC  DS    10XL128       -- RESOURCE CONTROL TABLE                 00056000
         ORG   GOVRESC                                                 00057000
GOVRESCT DSECT               -- RESOURCE CONTROL TABLE MAPPING         00058000
         DS    0D                                                      00059000
GOVOPTN  DS    CL16          ----- RESOURCE OPTION                     00060000
GOVNULLI DS    H             ----- INTEGER NULL INDICATOR              00061000
GOVPAD40 DS    CL2           ----- RESERVED FIELD                      00062000
GOVINTVL DS    F             ----- INTEGER OPTION REPRESENTATION       00063000
GOVNULLF DS    H            ----- FLOATING POINT NULL INDICATOR        00064000
GOVPAD50 DS    CL6          ----- RESERVED FIELD                       00065000
GOVFLOAT DS    D            ----- FLOATING POINT OPTION REPRESENTATION 00066000
GOVNULLC DS    H            ----- CHARACTER NULL INDICATOR             00067000
GOVCHLEN DS    H            ----- LENGTH OF THE CHARACTER OPTION       00068000
GOVCHAR  DS    CL80         ----- CHARACTER OPTION REPRESENTATION      00069000
GOVNEXTR DS    A            ----- POINTER TO NEXT RESOURCE CONTROL ROW 00070000
```

Figure 105. The DXEGOVA control block (Part 2 of 2)

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 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 DXEGOVA field GOVRESC. 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 the 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.

Figure 106 shows a part of the code for the IBM-supplied governor that 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    OH                -- 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 106. Resource initialization

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 on resource limits (contained in DXEGOVA) to determine when the resource limits are exceeded and when to cancel the user's activity.

For example, you can define a resource option that does not allow user JONES to use the EDIT TABLE command. You can then write your governor exit routine so that, if the XCBQRYP field of the DXEXCBA control block indicates an EDIT TABLE command, the governor exit calls the QMF cancellation service to cancel the command.

Table 56 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 QMF operating system you are running in.

Table 56. Fields of the DXEXCBA interface control block to the governor

Field	Data type	Purpose
XCBACTIV	CL1	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: 1 OPEN being run 2 FETCH being run 3 PREPARE being run 4 DESCRIBE being run 5 CLOSE being run 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.
XCBAIACT	CL1	Tells whether the current command is running interactively: 1 Interactive 0 Noninteractive (batch) 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. When running in New Function mode and the SQL authorization ID is larger than 8 characters, the value is truncated and placed in this field. See XCBAUTHX below for the complete SQL authorization ID.
XCBAUTHX	CL128	Contains the user's SQL authorization ID.

Table 56. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data type	Purpose
XCBCAN	CL1	Indicates whether the user or the governor requested cancellation of the current command. 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 XCBCMDP field. This field changes values 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. 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.
XCBDBMG	CL1	Identifies the database manager. This value is set to 2 for DB2 UDB for z/OS.
XCBEMODE	CL1	Indicates the current mode of the QMF session: 1 Interactive 2 Noninteractive (batch or server) 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. For more information about the command interface, see *Developing DB2 QMF Applications*.
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.
XCBKPARM	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 if 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.

Table 56. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data type	Purpose
XCBMGTXT	CL78	Contains the text for a message. The message can be logged in the QMF trace data, displayed on the screen, or both.
XCBMSGNO (ISPF only)	CL8	Contains the message ID for an ISPF message definition that can be used to log a message in the DSQDEBUG data set, viewed on your 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. (For a list of NLIDs used, see Table 1 on page ix.) Value does not change during a session.
XCBPANEL (ISPF only)	CL8	Contains the panel ID for the message Help panel for a cancellation message.
XCBPLAN	CL8	Contains the application plan ID for QMF. The 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 DBMS. The integer part of this decimal appears in the database status (“relative cost estimate”) panel. The value 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 `QMF V7R2.0`, and does not change during a session.
XCBQRYP	A	Contains the address of a copy of the query that QMF passes to the database for execution. 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. This field contains information only when data retrieval is requested through one of the following commands; no information is provided for queries z/OS DB2 system tables or QMF control tables. DISPLAY TABLE EDIT TABLE ERASE TABLE EXPORT TABLE IMPORT TABLE PRINT TABLE RUN QUERY SAVE DATA

Table 56. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data type	Purpose
XCBREFR	CL1	Indicates whether QMF refreshes the screen after returning from the governor; 1 indicates a refresh; 0 indicates no refresh. If your governor displays any screen information, set this field to 1.
XCBRELN	CL2	Identifies the QMF release level. For DB2 QMF Version 8.1, this is 13. The value does not change during a session.
XCBRGRP	CL16	Contains the name of the user's resource group. This value does not change during a session.
XCBROWSF	F	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. 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.
XCBSYST	CL1	Identifies the current operating system. The value does not change during a session, and is usually set to 3, indicating TSO, or native z/OS batch. Possible values are: 3 TSO, or native z/OS batch 5 CICS .
XCBTRACE	CL1	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). 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. For more information on tracing, see "Using the QMF trace facility" on page 338.
XCBUSER	CL8	Contains the user's TSO logon ID (for TSO), the user parameter on the job statement (for native z/OS batch). This field is not used in CICS; it contains blanks.
XCBUSERS	CL2048	Scratchpad area in which you can store results 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 illustrated below:

```
*************************************************************************** 00001000
*                                                                         * 00002000
*        CONTROL BLOCK NAME: DXEXCBA                                      * 00003000
*                                                                         * 00004000
*        FUNCTION:                                                        * 00005000
*                                                                         * 00006000
*            THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND           * 00007000
*            EXIT ROUTINES.                                               * 00008000
*                                                                         * 00009000
*        STATUS: VERSION 8 RELEASE 1 LEVEL 0                              * 00010000
*        (Version 8.1 applies only to OS/390 and z/OS                     * 00011000
*                 VM and VSE remain at Version 7.2)                       * 00012000
*        INNER CONTROL BLOSCK: NONE                                       * 00013000
*        CHANGE ACTIVITY:                                                 * 00014000
*                                                                         * 00015000
*                                                                         * 00016000
*************************************************************************** 00017000
*                                                                           00018000
DXEXCBA  DSECT                                                             00019000
         DS    0D                                                          00020000
XCBNAME  DS    CL8           -- CONTROL BLOCK IDENTIFICATION               00021000
         SPACE                                                             00022000
XCBEXCTL DS    XL190         -- EXIT CONTROL                               00023000
         ORG   XCBEXCTL                                                    00024000
XCBAUTH  DS    CL8           ----- AUTHORIZATION ID                        00025000
XCBUSER  DS    CL8           ----- USER ID                                 00026000
XCBPLAN  DS    CL8           ----- PLAN ID                                 00027000
         SPACE                                                             00028000
XCBQMF   DS    CL10          ----- CURRENT VERSION/RELEASE                 00029000
         SPACE                                                             00030000
XCBRELN  DS    CL2           ----- QMF RELEASE LEVEL                       00031000
         SPACE                                                             00032000
XCBTRACE DS    CL1           ----- QMF EXIT TRACE LEVEL                    00033000
XCBTOFF  EQU   C'0'          -------- NO TRACING                           00034000
XCBTPART EQU   C'1'          -------- PARTIAL TRACING                      00035000
XCBTFULL EQU   C'2'          -------- FULL TRACING                         00036000
         SPACE                                                             00037000
XCBSYST  DS    CL1           ----- OPERATING SYSTEM                        00038000
XCBSYSTX EQU   C'3'          -------- TSO,APPC, native                     00039000
XCBSYSTV EQU   C'4'          -------- CMS/VM/ESA                           00040000
XCBSYSTY EQU   C'5'          -------- CICS (OS/390 or VSE)                 00041000
         SPACE                                                             00042000
XCBPAD10 DS    CL4           ----- RESERVED FIELD                          00043000
         SPACE                                                             00044000
XCBNLANG DS    CL1           ----- CURRENT NATIONAL LANGUAGE               00045000
         SPACE                                                             00046000
XCBKPARM DS    CL1           ----- SETTING OF K PARAMETER                  00047000
XCBKPARN EQU   C'0'          -------- LATIN                                00048000
```

Figure 107. The DXEXCBA control block (Part 1 of 3)

```
XCBKPARY EQU   C'1'          -------- DBCS                                00049000
         SPACE                                                           00050000
XCBDBMG  DS    CL1           ----- DATA BASE MANAGER                      00051000
XCBDBMGS EQU   C'1'          -------- DB2 FOR VM/VSE                     00052000
XCBDBMGD EQU   C'2'          -------- DB2 FOR OS/390 and z/OS            00053000
XCBDBMGW EQU   C'3'          -------- WORKSTATION DB2                    00054000
         SPACE                                                           00055000
XCBEMODE DS    CL1           ----- CURRENT EXECUTION MODE                00056000
XCBIACTV EQU   C'1'          -------- INTERACTIVE MODE                   00057000
XCBBATCH EQU   C'2'          -------- BATCH MODE                         00058000
         SPACE                                                           00059000
XCBAIACT DS    CL1           ----- CURRENT INTERACT MODE                 00060000
XCBAIACY EQU   C'1'          -------- INTERACTIVE EXECUTION              00061000
XCBAIACN EQU   C'0'          -------- NOT INTERACTIVE EXECUTION          00062000
         SPACE                                                           00063000
XCBINCI  DS    CL1           ----- CURRENT COMMAND INTERFACE STATE       00064000
XCBINCIY EQU   C'1'          -------- COMMAND INTERFACE ACTIVE           00065000
XCBINCIN EQU   C'0'          -------- COMMAND INTERFACE NOT ACTIVE       00066000
         SPACE                                                           00067000
XCBINPRC DS    CL1           ----- PROCEDURE OR LIST CMD EXEC STATE      00068000
XCBPRCY  EQU   C'1'          ------- RUNNING A PROCEDURE OR LIST CMD     00069000
XCBPRCN  EQU   C'0'          ------- NOT RUNNING PROCEDURE OR LIST CMD   00070000
         SPACE                                                           00071000
XCBCVERB DS    CL18          ----- CURRENT COMMAND VERB                  00072000
         SPACE                                                           00073000
XCBCAN   DS    CL1           ----- CANCEL CURRENT COMMAND INDICATOR      00074000
XCBCANN  EQU   C'0'          -------- NO CANCELLATION                    00075000
XCBCANY  EQU   C'1'          -------- CANCELLATION IN PROGRESS           00076000
         SPACE                                                           00077000
XCBACTIV DS    CL1           ----- TYPE OF DATA BASE ACTIVITY            00078000
XCBOPEN  EQU   C'1'          -------- OPEN                               00079000
XCBFETCH EQU   C'2'          -------- FETCH                              00080000
XCBPREP  EQU   C'3'          -------- PREPARE                            00081000
XCBDESCR EQU   C'4'          -------- DESCRIBE                           00082000
XCBCLOSE EQU   C'5'          -------- CLOSE                              00083000
XCBEXEC  EQU   C'6'          -------- EXECUTE                            00084000
XCBEXECI EQU   C'7'          -------- EXECUTE IMMEDIATE                  00085000
XCBPAD20 DS    CL9           ----- RESERVED FIELD                        00086000
         SPACE                                                           00087000
XCBRGRP  DS    CL16          ----- RESOURCE GROUP NAME                   00088000
XCBPAD30 DS    CL22          ----- RESERVED FIELD                        00089000
         SPACE                                                           00900000
XCBCMDP  DS    A             ----- POINTER TO ORIGINAL COMMAND STRING    00091000
*                            -------- WILL NOT CONTAIN PROMPT VALUES     00092000
         SPACE                                                           00093000
XCBCMDL  DS    F             ----- ORIGINAL COMMAND STRING LENGTH        00094000
         SPACE                                                           00095000
XCBQCE   DS    F             ----- QUERY COST ESTIMATE VALUE             00096000
```

Figure 107. The DXEXCBA control block (Part 2 of 3)

```
SPACE                                                                  00097000
XCBROWSF DS    F            ----- DATA BASE ROWS FETCHED FROM SOURCE  00098000
*                           -------- SET BY QMF; EXIT MAY RESET       00099000
         SPACE                                                         00100000
XCBQERR  DS    CL1          ----- QMF ERROR INDICATOR                 00101000
XCBQERRN EQU   C'0'         -------- NO QMF ERROR DETECTED            00102000
XCBQERRY EQU   C'1'         -------- QMF ERROR DETECTED               00103000
XCBCLOC  DS    CL18         ----- CURRENT LOCATION NAME               00104000
XCBPAD40 DS    CL41         ----- RESERVED FIELD                      00105000
         SPACE                                                         00106000
XCBQRYP  DS    A            ----- POINTER TO SQL QUERY                00107000
*                           -------- QUERY LENGTH IS FIRST HALFWORD   00108000
         SPACE                                                         00109000
XCBUCTL  DS    XL432        -- USER CONTROL AREA                      00110000
         ORG   XCBUCTL                                                 00111000
XCBERRET DS    F            ----- EXIT ERROR RETURN CODE              00112000
XCBMGTXT DS    CL78         ----- EXIT ERROR MESSAGE TEXT             00113000
XCBMSGNO DS    CL8          ----- ISPF MESSAGE NUMBER                 00114000
XCBPANEL DS    CL8          ----- ISPF MESSAGE HELP PANEL             00115000
XCBLOGM  DS    CL1          ----- LOG MESSAGE INDICATOR               00116000
XCBLOGMN EQU   C'0'         -------- QMF SHOULD NOT LOG MESSAGE       00117000
XCBLOGMY EQU   C'1'         -------- QMF SHOULD LOG MESSAGE           00118000
XCBREFR  DS    CL1          ----- REFRESH SCREEN INDICATOR            00119000
XCBREFRN EQU   C'0'         -------- QMF DOES NOT HAVE TO REFRESH SCR 00120000
XCBREFRY EQU   C'1'         -------- QMF SHOULD REFRESH SCREEN        00121000
XCBPAD50 DS    CL28         ----- RESERVED FIELD                      00122000
         SPACE                                                         00123000
XCBUSERS DS    CL2048       -- USER SCRATCH PAD AREA                  00124000
XCBPAD60 DS    CL48         ----- RESERVED FIELD                      00125000
XCBAUTHL DS    H            -- LENGTH OF AUTHORIZATION ID             00126000
XCBAUTHX DS    CL128        -- AUTHORIZATION ID EXTENDED              00127000
XCBPAD70 DS    CL50         ----- RESERVED FIELD                      00128000
```

Figure 107. The DXEXCBA control block (Part 3 of 3)

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 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 IBM-supplied governor uses the code shown in Figure 108 on page 297 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, by replacing GOVUSERS in the following example with XCBUSERS. 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 108. Establishing addressability to the governor scratchpad area

Canceling user activity

When users reach their resource limits, you can call the QMF cancellation service to cancel user activity. For example, your governor exit routine might cancel the following:

- A QMF session during a function call at the start of a QMF session
- The current command during a number of different function calls, and any commands that start database activity

The code for canceling either of the first two activities is contained in the source program DSQU*n*GV1, DSQU*n*GV2, or DSQU*n*GV3. To have your governor call the QMF cancellation service to cancel an activity, branch to the address that appears in the DXEGOVA control block field named GOVCADDR. Figure 109 shows the statements that establish addressability to the QMF cancellation service. Before you use these statements to pass control from the governor exit routine to QMF, ensure that Register 13 points to a save area for the governor so that QMF can restore the state of the governor upon returning control.

```
L R15,GOVCADDR
         BALR R14,R15
```

Figure 109. Calling the QMF cancellation service

The cancellation routine returns control to the point addressed by Register 14 (in this case, the command that follows the BALR command). Register 15 contains a return code of 0 if QMF accepted the request to cancel, and a return code of 100 if the governor requested a cancel when QMF was inactive.

To cancel QMF commands using asynchronous processing in TSO or native z/OS, the IBM-supplied governor uses a timer macro, which returns control to a timer routine. The timer routine tests whether to cancel the current

command. If the command is to be canceled, it carries out the cancellation. The tests are based on processor time (TSO and native z/OS), and the number of rows fetched for the current DATA object. The tests can also be based on the user's response to a cancellation prompt.

The timer routine is the CSECT named TIMEX in the source code for the IBM-supplied governor. On z/OS the source code is the member DSQU*n*GV1 of the library QMF810.SDSQUSRE.

Making an asynchronous cancellation call is very much like pressing PA1. Cancellation might not be immediate, and it might be impossible. Before the cancellation takes place, control can return to the governor.

z/OS

Your governor exit routine can cancel asynchronous commands when a timer is active in TSO or native z/OS.

To cancel QMF commands using asynchronous processing in TSO or native z/OS, the IBM-supplied governor uses a timer macro, which returns control to a timer routine. The timer routine tests whether to cancel the current command. If the command is to be canceled, it carries out the cancellation. The tests are based on processor time and the number of rows fetched for the current DATA object. The tests can also be based on the user's response to a cancellation prompt.

Providing messages for canceled activities

Use this information to provide messages on z/OS.

z/OS

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:

XCBMGTXT
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, XCBMGTXT 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 message `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 preceding 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 IBM-supplied 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 long message text and can designate a panel ID. To use the long text for a message and the designated panel for Help, fill 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. For more information on the ISPF LOG service, refer to the appropriate *ISPF Dialog Management Services* manual.

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 in the U-setting of the trace option to get trace output. For additional details on using the QMF trace facility, see "Using the QMF trace facility" on page 338.

The IBM-supplied governor does not log messages for termination function calls. Messages do not appear on screen if the command is run in batch or noninteractively from a QMF application.

Upon entry to the governor, XCBMGTXT 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 message `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 preceding 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 IBM-supplied 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.

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 in the U-setting of the trace option to get trace output. For additional details on using the QMF trace facility, see "Using the QMF trace facility" on page 338.

The IBM-supplied governor does not log messages for termination function calls.

Assembling and link-editing your governor exit routine in TSO, ISPF, and native z/OS batch

Whether you are modifying the IBM-supplied governor exit routine or writing a routine of your own, you need to translate, assemble, and link-edit the routine. Use the sample link-edit statements shown in this section to help you.

Assembling your governor exit

QMF supports only assembler-language programming for a governor. This is the language, for example, in which the IBM-supplied governor is coded; the code was written for HLASM. You can review this code by printing certain members of the QMF810.SDSQUSRE library.

Link-editing your governor exit routine

Place the load module for the governor in a library available to all your QMF users. IBM recommends the library QMF810.SDSQLOAD, which contains the load modules for QMF itself. This library can be part of the concatenation for STEPLIB.

Name the module DSQU*n*GV1. These are the names of the IBM-supplied modules. Placing your own governor module in the QMF810.SDSQLOAD library replaces the IBM-supplied module, because that module is a member of that library.

To avoid replacing the IBM-supplied module, you can rename it or move it to another library. Or 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 QMF810.SDSQLOAD in the concatenation sequence. If it is not, QMF calls the IBM-supplied module instead of your own.

Be sure that the entry point for the new module is DSQU*n*GV1. If your source code begins with a CSECT statement with the DSQU*n*GV1 label, there is nothing extra to do. If your source code does not begin with the DSQU*n*GV1 label, specify the entry name on the END statement for the assembler code, or place it in an ENTRY statement in the linkage editor input.

Your own routine can run in either 31- 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 running in 24-bit mode and back to QMF in 31-bit mode.

```
ENTRY DSQUEGV1
   MODE AMODE(31),RMODE(ANY)
   NAME DSQUEGV1(R)
```

The QMF-supplied governor (DSQUEGV1) must run with AMODE(24) and RMODE(24).

```
ENTRY DSQUEGV1
   MODE AMODE(24),RMODE(24)
   NAME DSQUEGV1(R)
```

Assembling, translating, and link-editing your governor exit routine in CICS on z/OS

Whether you are modifying the IBM-supplied governor exit routine or writing a routine of your own, you need to translate, assemble, and link-edit the routine. Use the sample link-edit statements shown in this section to help you.

Translate your program 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).

Assembling your governor exit

QMF supports only assembler-language programming for a governor. This is the language, for example, in which the IBM-supplied governor is coded; the code was written for HLASM or Assembler H. You can review this code by printing certain members of the QMF810.SDSQUSRE library.

Link-editing your governor exit routine

Place the load module for the governor in a library available to all your QMF users. IBM recommends the library QMF810.SDSQLOAD, which contains the load modules for QMF. This library must be concatenated with DFHRPL in CICS.

Name the module DSQU*n*GV3. These are the names of the IBM-supplied modules. Placing your own governor module in the QMF810.SDSQLOAD library replaces the IBM-supplied module, because that module is a member of that library.

To avoid replacing the IBM-supplied module, you can rename it or move it to another library. Or you can place the module for your own governor in a different library in DFHRPL. For the last of these alternatives, be sure that your module's new library is ahead of QMF810.SDSQLOAD in the concatenation sequence. If it is not, QMF calls the IBM-supplied module instead of your own.

Be sure that the entry point for this module is DSQU*n*GV3. 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 for the assembler code, or place it in an ENTRY statement in the linkage editor input.

When link-editing, 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(ANY).

```
INCLUDE SYSLIB(DFHEAI)
   INCLUDE SYSLIB(DFHEAI0)
   ORDER DFHEAI,DFHEAI0
   ENTRY DSQUEGV3
   MODE AMODE(31),RMODE(ANY)
   NAME DSQUEGV3(R)
```

Using the DB2 governor on z/OS

DB2 has its own governor, which operates independently of the QMF governor. This section tells you what the DB2 governor does, and how you can use it for additional resource control. For more information on the DB2 governor, read the section on improving resource utilization in the *DB2 Universal Database for z/OS Administration Guide*. In DB2 publications, this governor is commonly called the Resource Limit Facility. You can control all access to the database and distributed access with the DB2 governor.

Monitoring the resources

The DB2 governor monitors the processor time consumed running certain queries. The queries it monitors are the dynamically run SELECT, INSERT, UPDATE, and DELETE queries. In a QMF session, this includes all queries of these types that are run in the following ways:

Using the QMF RUN command
The queries run might be SQL, QBE, or prompted queries. For QBE and prompted queries, the governor monitors the equivalent SQL queries.

Using other QMF commands
In support of other commands, QMF creates and runs SQL queries on behalf of the user. For example, among these queries are the SELECT queries that QMF runs in response to DISPLAY table commands. These commands do not present a prompt screen in response to +495 SQL codes.

Running the Table Editor
In support of the Table Editor, QMF creates and runs SQL queries on behalf of the user. For example, among these queries are the SELECT queries that QMF runs in response to SEARCH commands.

Differences between governors

You can supplement the operations of the QMF governor with the DB2 governor. Before you do, you should know how the governors differ.

- The DB2 governor limits its monitoring to the types of queries mentioned in the previous section. It does not monitor, for example, the processor time spent in running a CREATE or DROP query.
- 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.

When the maximum processor time is exceeded

When a query exceeds the maximum processor time, the DB2 governor ends the query and returns an SQL error code of -905. QMF then knows that the governor canceled the query. How QMF treats this information depends on where in a QMF session the 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 user's DSQDEBUG data set.

The session end might occur during periods when QMF sessions are not allowed. To enforce this restriction, people who attempt to use QMF during such a period of time might be assigned a maximum processor time of zero. This causes the cancellation of any monitored query.

After QMF Initialization

After initialization, QMF treats the cancellation of a query just as it treats any other error in running the query. Suppose, for example, that the governor cancels an INSERT query for which the user issued a RUN command. Then the inserts, if any, are undone, and the query panel is displayed with 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. Then 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.

Applying the DB2 governor to QMF

Before it can govern a QMF session, the DB2 governor needs input. Input in this case is the maximum processor time. The DB2 governor gets this input from a row in a resource limit specification table. In DB2 terminology, this table is an RLST. Such a table can be modified by anyone with appropriate DB2 authority (INSERT, UPDATE, and so on). By adding rows to one or more RLSTs, you can control the operation of the DB2 governor for your QMF users.

Selecting an RLST

Consider a DB2 subsystem into which QMF is installed. When the subsystem is started, it is associated with a specific RLST. This RLST then provides the DB2 governor input for all the subsystem users, including those who begin QMF sessions.

Different RLSTs can be associated at different times with the same DB2 subsystem. For example, your installation might use different RLSTs for different shifts. The RLST for one shift makes it impossible to use QMF during that shift. Any attempt to start a QMF session ends during QMF initialization, and a message appears in the DSQDEBUG data set.

Adding rows to an RLST

You (or someone with appropriate DB2 authority) can add rows to an RLST for your QMF users. A row contains:

- A value for the maximum processor time for column ASUTIME
- Specify value '2' for column RLFFUNC
- Specify collection name as 'Q' for column RLFCOLLN

For example, you might 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 blanks for the authorization ID.

Contact your DB2 administrator for information about what you can and cannot do with the RLSTs, and the structures of the tables. Each RLST has required columns with prescribed names and data types, but your installation might have added more columns. For general information on these tables, see *DB2 UDB for z/OS Administration Guide* .

Chapter 20. Running QMF as a batch program

If a user runs a procedure with the RUN command, he cannot execute QMF commands except to cancel the procedure or the session. Running a procedure using the RUN command might tie up considerable session time. Alternatively, given the proper authority, the user might run the procedure in batch mode. In this mode, the procedure runs independently of the user's session so that the user can continue to issue commands.

To enable your users to use batch mode, you must give them the proper authority. Your users can then use batch mode to run procedures independently of a session and issue commands interactively while the procedure is running. The batch procedure may not run immediately; it might wait to run after the user's QMF session ends.

You and your users can create batch procedures to be run and saved in the database. A procedure can invoke queries or other procedures and can execute other QMF commands. For more information about writing batch procedures, see *Using DB2 QMF*.

QMF also supplies the QMF BATCH application to simplify running batch jobs.

If you are using an NLF: Users at a multilingual installation can choose the language environment for their batch QMF sessions, just as they can for their interactive sessions.

Running QMF as a batch program on TSO/CICS

This chapter section how you can use the QMF batch mode in TSO, ISPF, native z/OS, or CICS. For ISPF on z/OS, the QMF batch facility executes QMF in the TSO terminal monitor program (TMP).

TSO

The order of information for z/OS is: TSO, ISPF, native z/OS, and CICS.

Authority to operate in batch mode (TSO)

To submit a batch job, you need to know what QMF and DB2 authority is needed.

How to determine the logon ID and DB2 primary authorization ID your job is running under:

- If your installation 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 installation does not use RACF, the logon ID and primary authorization ID are determined as described in "The PROFILE PREFIX statement" on page 311.

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.
- 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. For more information about which commands are appropriate for the batch environment, see *Using DB2 QMF*.

Users with authority to use QMF interactively, and who can run jobs in the background, can also use it in batch mode, while users who do not have authority cannot use it 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.

Sending a job to z/OS 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. For more information on writing procedures for batch, see *Using DB2 QMF*.

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 a FIB command requires the proper TSO authority. (Granting this

authority is a TSO administration task.) For more on FIB commands and their uses, see *TSO Extensions Command Language Reference*

The SUBMIT command can be run:
- During the user's QMF session by using the TSO command of QMF
- In TSO READY mode or in a CLIST that tailors the JCL of the job

The tailoring can be based on parameters whose values are passed to called CLIST.

Any error encountered while running a procedure can:
- Terminate the procedure
- Back out an uncommitted DB2 unit of recovery

The JOB statement for the job can specify that a message be sent to the user when the job is done. The message appears on the user's screen. The user need not end a QMF session to receive the message.

After the run is ended, the user can examine the printer output for errors. With the proper JCL, this output is routed to data sets that the user can examine with an editor. One of these data sets might contain a record of the confirmation and error messages and, if desired, a record of the QMF commands that have run.

JCL to execute a QMF batch job under TSO

Batch-job JCL is similar to a TSO logon JCL, because QMF is run in batch mode through batch-mode TSO. The JCL statements you can use in batch mode are discussed in this section.

The job statement: Begin your JCL with a JOB statement like this:

```
//BATCH   JOB USER=LMN,PASSWORD=ABC,NOTIFY=LMN
```

The statement shown might not be adequate for every installation, because it contains neither accounting information nor the user's name. The operands 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; among these are the MSGLEVEL and MSGCLASS operands that control the level of detail and the routing of the JCL and system messages.

Attention: 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, SYSTERM, 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 OUTPUT command of TSO (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.

SYSTSPRT: This data set contains the message output from TSO and ISPF. For this data set write:

```
//SYSTSPRT DD SYSOUT=A
```

SYSTSIN: SYSTSIN holds the TSO statements that run during the job step. To include these statements in your JCL, write the following:

```
//SYSTSIN DD *
   EXEC CLISTA
   PROFILE PREFIX(LMN)
   ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSMODE=B,DSQSRUN=LMN.PROCA)
⋮
/*
```

Figure 110. Adding the TSO statements from SYSTSIN

TSO runs these statements in their order of appearance in SYSTSIN:

- The first statement runs a CLIST named CLISTA, which might do allocations of QMF libraries.
- The second 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.

Where to place the statement: Place the PROFILE PREFIX statement before the first ISPSTART statement that starts QMF. Issuing the PROFILE PREFIX statement within QMF is ineffective.

How PROFILE PREFIX can change a profile:: By itself, the QMF SET PROFILE command makes no permanent changes to the user's QMF profile. In contrast, the PROFILE PREFIX statement can make permanent changes to the user's TSO profile, depending on your installation'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.

Making the PROFILE PREFIX effective: 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 running QMF in batch mode for the following reasons:

- To identify the user to QMF when RACF is not used

  At installations 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 the user in this case is the value specified by the DB2 installation parameter UNKNOWN AUTHID. The logon ID is used in trace output 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. See the discussion of this parameter in Chapter 10, "Customizing your start procedure," on page 79.
- 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, a procedure running under the logon ID LMN 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 export file name. Later, the IMPORT command imports this file.

If the user's dsname prefix is ABC instead of LMN, the file 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.

The procedure cannot find the file 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').

Running QMF batch in the foreground using TSO or ISPF: To start QMF in batch mode in the foreground, you can use any of the methods to start QMF that were discussed in Chapter 9, "Starting QMF," on page 67. For example, from the TSO READY mode, you can issue the following statement to start QMF from a CLIST:

```
ISPSTART CMD(clist__name) NEWAPPL
```

where `clist_name` is the name of the CLIST that starts QMF. This CLIST must contain a statement of the form:

```
ISPEXEC SELECT PGM(DSQQMFE) NEWAPPL(DSQE)
                    PARM(...DSQSMODE=B,DSQSRUN=aaa.bbb)
```

Here the ISPSTART statement runs in the foreground, not the background. You cannot do other things with TSO while waiting for the CLIST to end.

When the CLIST actually 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.

Debugging a procedure: You can use the trace codes and the HELP command to diagnose problems with a batch mode procedure. In fact, L2 tracing is the default for procedures run in batch mode. To change the trace setting, you need a SET command in 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 DSQDEBUG data set. 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 DSQDEBUG data set. You can then use the HELP command to display the corresponding message help panel.

Using the QMF batch query/procedure application (BATCH) in ISPF

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 the translated synonym to the users. They then issue the translated command synonym for BATCH. Refer to Chapter 16, "Customizing QMF commands," on page 197 for the procedure on how to assign synonyms.

Assigning authority to use the application on z/OS

Any QMF user can use the application, since starting it consists of running a shared procedure. The 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 SQL SELECT privilege.
- If the query or procedure results are to be saved in a new table, the user's SAVE command must be enhanced. (See "Enabling users to create tables in the database" on page 132.)

Using the application

Before starting the 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 fills in 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.

Starting the application

The 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 into a permanent data set
- Send the report to one or more other users

The advantage to using the application lies in its prompt panel, where the user outlines what the job should do and leaves the details of how to do it to the application. The user does not need to know JCL or QMF procedures.

To use the batch application, enter:

```
BATCH
```

which results in the display of the prompt panel in Figure 111 on page 315

Filling in the prompt panel

A user can get help filling out the prompt panel by pressing function key 1, which results in the display of the first of three help panels.

```
QMF BATCH              QUERY/PROC BATCH PROMPT

OBJECT NAME    ===>                                Name of query or procedure
Current OBJECT ===>  NO                       Use object in temporary storage?
QUERY or PROC  ===>  QUERY
PROC arguments ===>
FORM  NAME     ===>                                Form to be used with query
Current FORM   ===>  NO                         Use form in temporary storage?
BATCH NAME     ===>                           Name of QMF batch execution proc
DB2 SUBSYSTEM       ===>                        DB2 PLAN  ===>
LOGON PASSWORD      ===>                        TSO logon password
LOGGING             ===>  YES                   Log messages and commands?
SAVE DATA           ===>                        Name of data to be saved
REPORT DATASET      ===>
   NEW DATASET      ===>                        Is the data set new?
   VOLUME           ===>                        Optional if NEW or uncataloged
   REPORT WIDTH     ===>  133                   Width of report line
VIEW REPORT         ===>  YES                   Should report be printed?
OUTPUT CLASS        ===>  A                     Class for PRINT and TRACE
DISTRIBUTION        Enter userids and nodes to send report.
      USERID        ===>                  NODE     ===>
                    ===>                           ===>

PF1=Help PF3=End  Enter=Process batch request
```

Figure 111. The QMF batch prompt panel

Required entry fields: Certain fields on the batch prompt panel are mandatory. Messages are displayed prompting the user to enter values for these required fields if the Enter key is pressed before values are provided. The cursor is then positioned on the field requiring input. Table 57 describes the required fields.

Table 57. BATCH application required entry fields

Field	Description
OBJECT NAME	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.
QUERY or PROC	The object type to be run in batch; must be either QUERY or PROC.

Table 57. BATCH application required entry fields (continued)

Field	Description
BATCH NAME	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 BATCH NAME field for each query or the new batch job replaces the old job. This procedure contains the appropriate QMF commands depending upon the user's input. The user's query or procedure, specified in the 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: Table 58 describes the remaining (optional) entry fields on the panel. Where a value of YES or NO is expected, a default YES or NO normally appears on the screen. If a user blanks out a value in a YES/NO field, the user is prompted for an entry.

Table 58. BATCH application optional entry fields

Field	Description
Current OBJECT	If the batch query or procedure is currently in temporary storage, the user enters 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.
Arguments to the REXX procedure named in the OBJECT NAME field.	
PROC ARGUMENTS	Through this field, you can pass arguments to the REXX procedure specified in the OBJECT NAME field.

Table 58. BATCH application optional entry fields (continued)

Field	Description
FORM NAME	To run the batch query using a form, the user must specify the name of a form in this field. If the form to be used: • Is the default form, leave the field empty. • 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. • 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. This form is saved using CONFIRM = NO as a profile setting. If you enter the name of an existing form, the new form replaces the old.
Current FORM	If the batch form is the current form, the user enters 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.
DB2 SUBSYSTEM	Enter the name of the DB2 subsystem that QMF uses; it has the same value as program parameter DSQSSUBS.
DB2 PLAN	Enter the name of the QMF application plan; it has the same value as program parameter DSQSPLAN.
LOGON PASSWORD	Enter your logon password; it does not appear on the screen.
LOGGING	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 the user does not want tracing at the L2 level, NO should be specified. Tracing does not continue in the batch procedure beyond the SET PROFILE (TRACE=NONE command, which is then in the generated user procedure.
SAVE DATA	If the user wants the data resulting from running a query or procedure to be saved, a value must be given for this field. The DATA is saved as a new table using this name and the CONFIRM=NO keyword option.

Table 58. BATCH application optional entry fields (continued)

Field	Description
REPORT DATASET	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. This data set name is passed to z/OS JCL 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, @, #, $. If quotation marks are used, z/OS assumes that special characters are used and does not catalog the data set in the system catalog.
NEW DATASET	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.
VOLUME	You can optionally fill this field if you entered YES in the NEW DATASET 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.
REPORT WIDTH	If you entered YES in the NEW DATASET field, you must fill in 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.
VIEW REPORT	This field must contain YES or NO. YES indicates print the job; NO indicates do not print the job.
OUTPUT CLASS	Enter the output class for the printed output from your job. The printed output includes: • The system messages • The report (DSQPRINT), if it was printed • The trace output (DSQDEBUG) • An abend dump (DSQUDUMP), if one was produced If your installation provides for it, you can choose an output class that holds the printed output for routing to your terminal.

Table 58. BATCH application optional entry fields (continued)

Field	Description
DISTRIBUTION USERID and NODE	If the user wants the resulting report to be sent to other users, the user must 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 REPORT DATASET field. 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 need not fill in the NODE field for a given user if that information is in your NAMES.TEXTLIST data set. The node ID you write might correspond to an entire list of names in this file, allowing you to send the report to more than just two people.

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 you do not risk losing your changes when maintenance is performed.

The 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 QMF810.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 the members DSQBE00, DSQBE01, and DSQBE02 of the QMF810.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 QMF810.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-tailoring models in the QMF810.SDSQSLBE library:

- DSQABB1J models the JCL for the batch job. This models a procedure that runs a query in batch mode.
- DSQABB1P and DSQABB1S model QMF procedures. They model 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 model file DSQABB1J 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 name DSN. If you want QMF to run in a DB2 subsystem with a different name, add DSQSSUBS=xxx to the PARM operand of the ISPSTART command (where *xxx* is the appropriate subsystem name).
- Allow the user to specify a GDDM nickname for the printed report.
- Add extra logic to enforce your installation'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 installation.

  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.

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 is maintained, 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.

Modify the batch application with all users having the same PROFILE PREFIX, and assume that all users have unique user IDs. Add the user IDs to the data set names using &SYSUID and &ZUSER.

You need to make three modifications to DSQABB1S SKELETON. Figure 112 shows the required changes. 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 OS/390 BATCH
)CM                    RESETS THE PROC ITEM
)CM                    FREES ISPFILE 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(ISPFILE) DELETE
&RESET &PROC
)CM &IMPORT &PROC &FROM DSQABB
&IMPORT &PROC &FROM &ZUSER..DSQABB

)SEL &ITM = &QUERY
&DISPLAY &QUERY
)ENDSEL
```

Figure 112. Modifying the DSQABB1S SKELETON

Make the five modifications to DSQABB12 CLIST as commented in Figure 113 on page 322.

```
/*************************************************************/   00088000
/*  ALLOCATE USERID.DSQ1EBFT.PROC TO BE USED FOR ISPF        */   00089000
/*  FILE TAILORING OUTPUT.                                   */   00090000
/*************************************************************/   00091000
 FREE FILE(ISPFILE)                                               00092000
/* ALLOC DDNAME(ISPFILE) DSNAME(DSQ1EBFT.&PROC) OLD              00093000
 ALLOC DDNAME(ISPFILE) DSNAME(&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(ISPFILE) DSNAME(DSQ1EBFT.&PROC) NEW SPACE(5,2) + 00098000
/* TRACKS DIR(10) USING(ATTRPDS) CATALOG                         00099000
   ALLOC DDNAME(ISPFILE) DSNAME(&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
⋮
```

Figure 113. Modifying DSQABB12 CLIST (Part 1 of 2)

```
/*************************************************************/         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(ISPFILE) 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 113. Modifying DSQABB12 CLIST (Part 2 of 2)

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 the JCL shown in Figure 114 on page 324 to run QMF as a batch job in native z/OS.

```
/*************************************************************/      00299000
//QMFBAT   JOB                                                       00300000
//S1       EXEC PGM=DSQQMFE,PARM='M=B,I=yourQMFproc'                 00301000
//*                                                                  00302000
//*  Program libraries required when running in batch                00303000
//*                                                                  00304000
//STEPLIB  DD  DSN=QMF810.SDSQLOAD,DISP=SHR                         00305000
//         DD  DSN=DSN.SDSNEXIT,DISP=SHR                            00306000
//         DD  DSN=DSN.SDSNLOAD,DISP=SHR                            00307000
//         DD  DSN=GDDM.ADMLOAD,DISP=SHR                            00308000
//*                                                                  00309000
//*  QMF/GDDM maps are required when running in batch                00310000
//*                                                                  00311000
//ADMGGMAP DD  DSN=QMF810.SDSQMAPE,DISP=SHR                         00312000
//*                                                                  00313000
//*                                                                  00314000
//*  Datasets used by QMR                                            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
//DSQUDUMP DD  SYSOUT=A,DCB=(RECFM=VBA,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 114. JCL to run QMF as a native z/OS batch job

When you run QMF in native z/OS, remember these facts:

- TSO is not available.
- QMF functions that require TSO or ISPF will 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 name to export or import files.
- You cannot use procedures with logic (REXX PROCS). To run QMF with REXX in a non-TSO address space, you must use IRXJCL, as illustrated in Figure 115 on page 325.

The REXX program listed in Figure 115 on page 325 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=DSN.DB2A.SDSNLOAD,DISP=SHR
//         DD  DSN=DSN.DB2A.SDSNEXIT,DISP=SHR
//         DD  DSN=QMFDEV.QMF810.SDSQLOAD,DISP=SHR
//ADMGGMAP  DD  DSN=QMFDEV.QF720.SDSQMAPE,DISP=SHR
//SYSEXEC   DD  DSN=ROBIN.QMF810.SDSQEXCE,DISP=SHR
//DSQPRINT  DD  SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=1330)
//DSQDEBUG  DD  SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//DSQUDUMP  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,(1,1),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
IF DSQ_RETURN_CODE = DSQ_SEVERE THEN EXIT DSQ_RETURN_CODE
CALL DSQCIX "EXIT"
SAY DSQ_MESSAGE_ID DSQ_MESSAGE_TEXT
EXIT DSQ_RETURN_CODE
/*
```

Figure 115. REXX program to start and run QMF in batch mode

Running QMF as a non-interactive transaction on CICS

QMF runs interactively as a conversational transaction. 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 noninteractively.

The QMF transaction can be run from a terminal or as a transaction running without a terminal.

Running batch from a terminal

You can run QMF from a terminal to produce a report. For example, you can write the procedure in Figure 116 on page 326 to produce a report located in CICS auxiliary storage. (QMF treats lines that begin with "--" as comments in QMF 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 (QUEUENAME=STATRPT1,QUEUETYPE=TS)
   --
   -- End of procedure
```

Figure 116. Producing a report located in CICS auxiliary storage

Execute the QMF transaction as described here to run this procedure in batch mode:

```
QMFE M=BATCH,I=STATRPT1__PROC
```

QMF runs this transaction without displaying any screens. Upon successful completion of the procedure, the report is located in CICS storage queue STATRPT1. You can then view the report using the CICS-supplied transaction CEBR:

```
CEBR STATRPT1
```

Running batch without a terminal

A QMF transaction can also be run without a terminal. A terminal used to run a batch job is locked until QMF completes the transaction. To run a QMF procedure in batch mode without a terminal, use the EXEC CICS START command. The following example runs the QMF procedure STATRPT1__PROC:

```
EXEC CICS START TRANSID(QMFE) FROM(M=BATCH,I=STATRPT1__PROC)
```

When this transaction completes, the CICS storage queue STATRPT1 can be browsed using the CICS-supplied transaction CEBR.

Debugging a procedure

QMF provides a command-level and message-level trace facility. This facility is useful when there is a problem running a QMF procedure in batch mode. QMF command-level and message-level tracing is automatically active when running QMF in batch mode. You can route this message trace to CICS auxiliary temporary storage or the transient data queue.

For example, to run the previous 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=BATCH,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.

When routing QMF trace to CICS auxiliary storage, do not set full component-level trace; temporary storage will fill up quickly. Transient data (the default) is recommended when doing other than message-level tracing.

Termination return codes

Termination return codes for QMF are **0**- normal termination, and **8**- abnormal termination.

Chapter 21. Troubleshooting and problem diagnosis

Use this chapter to help solve problems your users might have while using QMF. "Troubleshooting common problems" provides possible solutions to common problems, while "Determining the problem using diagnosis aids" on page 336 provides explanations of diagnosis aids that help you solve more complex problems.

Troubleshooting common problems

Use this section to help determine how to solve initialization errors, printing errors, warning messages on the display, incoherent report displays, and slow response times or other performance problems.

Handling initialization errors

If you cannot start QMF, there are several common fixes:

- Determine if all QMF users at your shop cannot get into QMF, or is it just one user.
- Check whether there are any messages on the terminal screen, and look up the explanation for the DSQDEBUG file message in *DB2 QMF Messages and Codes*.
- If nothing appears on the screen and nothing is in DSQDEBUG, go into ISQL and issue a SELECT * FROM Q.ERROR__LOG and see if 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 book for the appropriate product.

  Check that the DB2 database is initialized and working properly. If all users are getting a type of ADMxxxx message upon start up, check that the base GDDM product is working correctly by running the GDDM IVPs.

z/OS concerns

If any DSN (DB2) and ADM (GDDM) error messages appear, look them up in *DB2 QMF Messages and Codes*.

Users should still look on the screen for more messages, and in DSQDEBUG and Q.ERROR__LOG for more information. If there are no other messages, have the user try to run the TSO command PROFILE MSGID WTPMSG and restart QMF.

Handling warning messages

If errors occur during QMF initialization (or after issuing the CONNECT command), you might see this message on the QMF Home panel:

```
Warning messages have been generated
```

Errors that cause this kind of message do not stop QMF. They indicate that QMF is having a problem loading or reading any of the following:

- Command synonym table
- Function key definitions table
- Resource control table (for governor exit routine)
- User edit exit routine
- Governor exit routine
- Module level trace control

For command synonyms, function keys, and resource control tables, ensure that:

- The user has the SQL SELECT privilege for that 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 Chapter 16, "Customizing QMF commands," on page 197
  - The structure for function key tables is shown in Chapter 17, "Customizing QMF function keys," on page 213
- All rows of the table contain valid data. If this might be the problem, see:
  - "Entering command synonym definitions into the table" on page 202 for information on valid command synonym definitions
  - "Entering your function key definitions into the table" on page 217 for information on valid function key definitions
- All rows in the tables are unique.

To view the information in the trace data, first press the Help key to display a panel containing the message number. Then browse or print the user's trace data. Search the trace data for the numeric portion of the message number to see information about the error.

z/OS concerns

More information about the error is logged in the user's trace data. In TSO 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 using the DSQSDBQT or DSQSDBQN program parameter when you started the QMF session.

Handling GDDM errors during printing

If a GDDM error occurred during printing, 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 section discusses some common errors and what you can do to fix them.

DSQ50623

GDDM error. ADM0307 E FILE 'ADMPRINT.REQU—QUEUE' NOT FOUND. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, 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.

DSQ50623

GDDM error. ADM0314 E UNABLE TO OPEN 'MYPRINT'. DD STATEMENT MISSING. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, QMF was able to find a DD statement for the output. You need to provide a DD statement to your QMF startup EXEC, CLIST, or JCL to specify what to do with output from the nickname.

DSQ50623

GDDM error. ADM0482 E DEVICE NAME LIST '31E' IS INVALID FOR FAMILY 1. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, 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. For a list of valid device tokens for each family of GDDM printers, see *GDDM System Customization and Administration*.

DSQ50631

GDDM error. ADM0904 E ALPHANUMERIC FIELDS ARE NOT SUPPORTED FOR THIS DEVICE. Severity 8. Function ASDFLD. * CMD=PRINT**

If you see a message like this, 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. For more information on what families of printers handle your type of output, see *GDDM System Customization and Administration*.

DSQ90551

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 defaults module.

DSQ50633

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. Contact your CICS administrator for help with this problem (either modifying the JCL and restarting CICS or opening the queue).

Handling GDDM errors on z/OS

DSQ50623

GDDM error. ADM0307 E FILE 'ADMPRINT.REQU—QUEUE' NOT FOUND. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, 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.

DSQ50623

GDDM error. ADM0314 E UNABLE TO OPEN 'MYPRINT'. DD STATEMENT MISSING. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, QMF was able to find a DD statement for the output. You need to provide a DD statement to your QMF startup EXEC, CLIST, or JCL to specify what to do with output from the nickname.

DSQ50623

GDDM error. ADM0482 E DEVICE NAME LIST '31E' IS INVALID FOR FAMILY 1. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, 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. For a list of valid device tokens for each family of GDDM printers, see *GDDM System Customization and Administration*.

DSQ50631

GDDM error. ADM0904 E ALPHANUMERIC FIELDS ARE NOT SUPPORTED FOR THIS DEVICE. Severity 8. Function ASDFLD. * CMD=PRINT**

If you see a message like this, 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. For more information on what families of printers handle your type of output, see *GDDM System Customization and Administration*.

DSQ90551

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 defaults module.

DSQ50633

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. Contact your CICS administrator for help with this problem (either modifying the JCL and restarting CICS or opening the queue).

Handling QMF errors during printing on z/OS

The information in the following table helps you to solve errors that can occur during printing:

What happens	What it means	What to do
You issue the PRINT command from the command line or a function key and see the message: `GDDM printer nickname is required for PRINTER.`	The object you are trying to print needs a printer name, and no printer name default 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 printer name default 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.

What happens	What it means	What to do
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: • For a report, table SQL or QBE query, procedure, or profile, specify PRINTER=' ' to bypass GDDM printing. • For other objects, use `PROCOPT=((PRINTCTL,0))` with no device token.
When printing a report, table, SQL or QBE query, procedure, or profile, you see the message: `File DSQPRINT did not open.`	No printer name default exists in your profile, and no DSQPRINT data set or system output is currently allocated.	Allocate DSQPRINT before issuing a print command.

Reminder: 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)
```

Handling display errors

If a user who attempts to display a report finds that the 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. QMF provides three ways of handling these control characters:

- Using the hex function
- Using the QMF-provided hex and bit edit codes in the QMF form
- Handling binary data through user-written edit routines

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 QMF-provided HEX and bit edit codes

Two edit codes (and their wrapping versions) allow QMF to display binary data in character columns: X and XW (for HEX display), B and BW (for bit display). For more information on using these edit codes, see *DB2 QMF Reference*.

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. For example, assume that each bit represents a data item and displays in Natural Language Form of the value. If the fifth bit represents gender rather than HEX values, a user edit code routine can cause a value of `Male` Or `Female` to be displayed.

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 you create. See Chapter 18, "Creating your own edit codes for QMF forms," on page 227 for more information.

Solving performance problems

If your users notice slow performance in running queries or formatting reports, the problem might be that QMF is attempting to retrieve all the database rows requested during one command before starting another. It is also possible that the user does not have enough virtual storage to retrieve all the requested rows. This section explains what you can do to solve each kind of problem.

Increasing the user's report storage

Users might also experience slow performance if they do not have enough virtual storage to accommodate a large report. 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 user memory. The user might find performance slow for formatting complex reports or scrolling the report.

To maximize report performance, make sure 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 have.

You can also define a spill file for the user. 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. For CICS, because a spill file can hold a maximum of 32, 767 rows of size 4K each, setting DSQSBSTG higher ensures that QMF will complete the report.

Even with a spill file, a user can encounter the incomplete data condition. If this occurs often, you might want to find if there is an additional problem.

QMF performance may 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.

z/OS concerns

Setting the DSQSRSTG parameter at a very high value can also cause slow performance.

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 query. For a description of this query, see *DB2 UDB for z/OS SQL Reference*.

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. These recommendations are in addition to any storage required by additional products installed.

Determining the problem using diagnosis aids

If you were not able to solve your problem using the troubleshooting techniques discussed in "Troubleshooting common problems" on page 329, use this section to find out which QMF and TSO diagnosis aids can help you to determine the problem.

Choosing the right diagnosis aid for the symptoms

Use Table 59 to help you determine which diagnosis aids you need for the symptoms you are experiencing. The diagnosis aids are listed across the top of the table, and symptoms are listed on the side. For example, if you experience a problem while using a governor exit routine, you can use the QMF trace facility, CICS or TSO status information, and QMF messages and help to determine the problem.

Table 59. Types of problems and the best diagnosis aids to use for them

	QMF msg. no.	QMF trace	Dump	Status info	Help message	Non-QMF msg. no.	Error log output
Abend	X		X	X			
Batch session	X	X		X		X	X
Callable interface	X	X	X	X		X	
Display panel	X	X			X	X	X
Document interface	X	X			X	X	X
Error messages	X	X			X	X	X
Governor exit routine	X	X	X	X	X	X	
Incorrect output	X	X			X	X	X
Initialization	X	X		X	X	X	X
Installation	X				X	X	X
Interrupt facility	X	X			X	X	X
Loop		X		X		X	X

Table 59. Types of problems and the best diagnosis aids to use for them (continued)

	QMF msg. no.	QMF trace	Dump	Status info	Help message	Non-QMF msg. no.	Error log output
Performance	X	X		X		X	X
Printing	X	X		X	X	X	X
QMF command	X	X			X	X	X
SQL error codes	X	X			X	X	X
Termination	X	X		X	X	X	X
User edit routine	X	X	X	X		X	X

Diagnosing your problem using QMF 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. These numbers are listed in *DB2 QMF Messages and Codes*, which provides more information on how you can solve the problem.

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 Support Center representative might need this number. To make sure the number displays, set the global variable DSQDC__SHOW__PANID to 1:

```
SET GLOBAL (DSQDC__SHOW__PANID=1
```

Determining which QMF function issued an error message

You can use the QMF message number, which begins with DSQ, to determine which QMF component issued the message. This information can help you isolate the problem to a specific QMF function.

The QMF functions and their associated ranges of message numbers are shown in Table 60. The trace IDs are the same IDs that you use to trace QMF activity for each function.

Table 60. QMF functions and the message numbers they issue

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

Table 60. QMF functions and the message numbers they issue (continued)

Function	Trace ID	Message numbers
Report formatting	F	DSQ60000 - DSQ69999
Charting	P	DSQ70000 - DSQ79999
Full-screen windows	G	DSQ80000 - DSQ89999

In addition to the message numbers in Table 60 on page 337, the following ranges of message numbers might be generated during QMF initialization:

DSQI0001 - DSQI0100
DSQ90000 - DSQ99999

Handling system error messages

A system error might indicate a system problem, a resource problem, or an unexpected condition. These might be problems within QMF, the database manager, or possibly 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, or see *DB2 QMF Messages and Codes*.

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.

Handling SQL return codes

In some cases, the message QMF displays might map to an SQL return code. For example, suppose a user receives QMF message DSQ10422. This message maps to the SQL return code -30060, which has the text:

```
RDB AUTHORIZATION FAILURE
```

See *DB2 Messages and Codes* for the SQL return codes.

Using the QMF 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. This section shows you how to allocate storage for the trace output, how to start the facility and determine the level of tracing detail, and how to view the trace data for diagnosis.

The trace facility on z/OS

Follow these instructions for using the trace facility on z/OS.

Allocating the trace data set (TSO): Certain procedures in this book rely on abend information as well as trace information that QMF records in the DSQDEBUG data set.

Allocating for TSO or native OS/390: Trace information is recorded in the DSQDEBUG data set. You can find abend dump information in the DSQUDUMP and SYSUDUMP data sets. Make sure that these data sets are allocated before you begin the QMF session. The data sets are automatically allocated by the LOGON procedure for the user ID under which you intend to operate.

Check with TSO administration if you are not sure whether these data sets are automatically allocated before a QMF session. If they are not, 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(DSQUDUMP) SYSOUT(A) USING(DUMP)
ALLOC DDNAME(SYSUDUMP) SYSOUT(A)
```

Figure 117. Allocating the data sets for TSO

Allocating for CICS: The trace is recorded in the DSQDEBUG data set. This data set should be allocated in the CICS startup JCL. The trace can be shared between all users in the same CICS address space.

Starting the trace facility:

1. Allocate a data set with a *ddname* of DSQDEBUG.

   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.
2. Decide on your tracing options.

   With these options, you control what is traced and the level of detail.

   Specify a value of ALL on the DSQSDBUG program parameter when you start QMF. This value traces QMF activity at the highest level of detail, including program failures that might occur during QMF initialization.

   You need to use a transient data queue to hold the output if it exceeds 32,767 rows.
3. Specify these options to QMF Trace.

   During a QMF session, some set of tracing options is always in effect. You can override current trace options in several different ways:

- Instruct the user to enter the following QMF command:

  ```
  SET PROFILE (T=value
  ```

  where `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.
- Users can do a DISPLAY PROFILE to change the TRACE parameter in the profile. If the user wants to make this setting permanent (until the next change), he can hit PF2 to save it.
- Users can issue setting, SET (T=value. This setting will temporarily change the user's profile. To save this setting, the user can issue the SAVE PROFILE command.

4. Access the trace data set when you have a warning or a system error during QMF initialization.

   Looking at DSQDEBUG helps you understand the reason for the error.
5. Interpret the trace output.

   You can display or print the DSQDEBUG file for analysis.

Getting the right level of detail in your trace output: If you want to trace all QMF functions at the most detailed level, use a value of ALL for the trace.

If you want to trace individual QMF functions, update the TRACE column of Q.PROFILES with a character string that has letters for the QMF functions you want to trace and numbers for the level of detail you want in the trace data for each function. You need to pair each letter with a number:
The value 1 traces a function at a medium level of detail.
The value 2 traces a function at the highest level of detail.

Only the functions you specify in the character string are traced. The letter for each QMF function is shown in the following list.

Trace ID	QMF Function
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 list

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
```

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 issued by QMF; L2 writes all the L1 records, plus additional records describing the execution of QMF commands. Use the L2 trace code to log each command a user issued and how QMF responded to that command. Figure 118 shows an example of a RUN QUERY command that failed because the user named columns that were 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.
&O1:  DATE
&O2:  STAFF
&O9:  -205
------------------------------------------------------------------------
```

Figure 118. 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 shown in Figure 118 on page 341 appears in *DB2 QMF Messages and Codes* as:

```
Column &01 is not in table &02.
```

The bottom half of Figure 118 on page 341 shows that the value for &01 in the message is DATE and that the value for &02 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 associated with the error message. Use the variable values from the trace data together with the help command to reconstruct the message help panel.

Tracing at the module level: Important: Perform a trace at the module level only under IBM Service Level 2 guidance.

You can turn on a trace for certain modules using the SET PROFILE command and the module DSQUTRAC. 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:

The value 3 provides a detailed trace for specific programs in a component, and traces entry and exit for all other programs in the component.
The value 4 traces a module only.

To create a module-level trace, list the modules you want traced in the DSQUTRAC module. 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
```

Viewing QMF trace data: DSQDEBUG holds the information recorded by the trace facility. It must be allocated before you start QMF if tracing is to be used. 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 very long.

Printing or displaying in TSO: The DSQDEBUG data set might have been allocated automatically through your LOGON profile in a TSO environment.

Even so, you can reallocate it if the original allocation does not fill your needs (for example, the original allocation might define DSQDEBUG as a PRINT file when you really want to display it).

To allocate (or reallocate) for printing, issue the following statements, 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)
```

The allocation contains 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.

Reminder: 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) DSQDEBUG as a sequential data set that can be displayed using an online editor. The data set consists of 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 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. In CICS, the trace data set is described as an extra region data set. The trace data set is described in CICS tables by a DCT TYPE=SDSCI command and a DCT TYPE=EXTRA command, as in Figure 119 on page 344.

```
* TRACE DATA SET
       DFHDCT TYPE=SDSCI,DSCNAME=DSQDEBUG,
              RECFORM=VARBLK,
              RECSIZE=121,
              BLKSIZE=6050,
              TYPEFILE=OUTPUT
 *
 *
       TITLE 'DSQDCT - CICS DESTINATION CONTROL TABLE'
 *
 * TRACE DATA SET
 *
 DSQD DFHDCT TYPE=EXTRA,DESTID=DSQD,DSCNAME=DSQDEBUG,RSL=1
```

Figure 119. Description of the trace data set in the CICS environment

QMF trace data from all the QMF users in a single CICS region are written to a single trace data set. Each trace entry contains the terminal ID of the user that recorded it.

To look at the trace data set while the CICS region is active, you must close the trace data set using the CICS queue ID DSQD. You can use this ID while using the CICS-supplied transaction CEMT. After the trace data set is closed, you can print or browse it.

While the trace data set is closed, no other records are written by CICS users. In this state, QMF continues to operate without recording trace records. To make the QMF trace available again, you can use the CICS-supplied transaction CEMT to open the trace data set using the CICS queue ID DSQD.

Determining the QMF service level: The service level information is displayed:

- When T=ALL is specified on invocation (or from Q.PROFILES)
- When SET (TRACE ALL was specified as a command
- When an abend occurs

You can determine the QMF service level using the following procedure:

1. Enter the SET PROFILE command (T=ALL.
2. Enter the SET PROFILE command (T=NONE.
3. Exit QMF.
4. Look at the DSQDEBUG file.

The resulting trace shows the program with its version, date, and time. The trace can also show an Authorized Program Analysis Report (APAR) number if the module has a Program Temporary Fix (PTF) applied, as in the following trace example:

```
** DSQFQWRM: ENTERED FROM DSQFMCTL ***
   V7R2.00   00/01/30 12:00 PNxxxxx
```

APAR PNxxxxx is the most recent APAR for which service was 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.

To turn tracing off, issue the following command from within QMF:

```
SET PROFILE (T=NONE
```

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 DSQSDBUG controls this tracing when QMF is started.

Diagnosing abends

You might need to diagnose abends using diagnostic facilities in TSO, z/OS, or CICS facilities available in your environment. (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. Figure 120 shows an example.

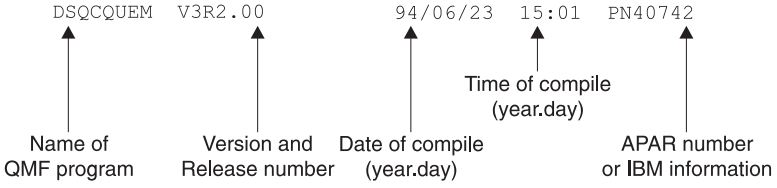


Figure 120. Example of a stamp that identifies a QMF program

Using z/OS diagnostic facilities

To diagnose an abend, you might need to use procedures in the appropriate *Tools and Service Aids*, 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 to the operating system, and allows it to continue with the abnormal termination process.

If an abend occurs while processing the user edit code or while executing the governor, additional areas appear in the dump to assist with problem diagnosis.

For the user edit code, DXEECS, the input area, and the result area are added to the output.

For the governor, DXEXCBA and DXEGOV are added to the output.

Using CICS diagnostic facilities

To diagnose an abend in QMF, you might need to use procedures in the *CICS Problem Determination Guide*. Because another program might have caused QMF to abend, these procedures 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 chapter do not contain enough information 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 terminal 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. Figure 121 on page 347 shows an example of diagnostic display attributes you might use.

The definition shown in Figure 121 on page 347 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 time-out, 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
   ERRHilight       : Underline        No | Blink | Reverse | Underline
```

Figure 121. TYPETERM specification for CICS diagnostic display

Using the QMF interrupt facility

z/OS

In TSO, the QMF interrupt handler can be activated even though a QMF command is inactive. To interrupt QMF, press the PA1 key. You need to refresh the screen to see the QMF procedure panel. To do this, press the PA2 key.

Use the QMF interrupt facility to gather information about a problem. Using the interrupt facility, you can produce an abend dump, or cause trace information to be displayed or written into the DSQDEBUG data set.

You use the interrupt facility under the logon ID of the user whose problem you are diagnosing. However, you must recreate the problem first, unless you were there when it occurred.

Creating an interrupt: The first step in using the interrupt facility is to create an attention 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. If these combinations do not work for you, see the appropriate publications for your current system configuration to obtain more information on creating the interrupt.

The interrupt facility responds by displaying the following message:

```
DSQ50546 QMF command interrupted!    Clear screen and press enter.
```

Figure 122. QMF interrupt handler prompt 1

Displaying trace information after creating an interrupt: After the interrupt message appears, press the Clear and Enter keys, 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 123. QMF interrupt handler prompt 2

Make your choice by typing CONT, CANCEL, or DEBUG, then press the Enter key:

- Enter CONT to return control to wherever you were before you caused the interrupt, as if the interrupt had 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. Note that it is not always possible to cancel a command.
- Enter DEBUG to get diagnostic information as shown in Figure 124

```
-- 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 124. Diagnostic information captured by typing DEBUG on the interrupt screen.

The trace information on the second line of this example tells you that, at the time of the interrupt, control was in CSECT DSQEINPT, and that control had reached this CSECT by passing successively through the CSECTs DSQDSUPV, DSQDSUPX, DSQEADAP, and DSQEMAIN.

Respond to the debug panel shown in Figure 124 by entering CONT, CANCEL, ABEND, TRACEALL, or TRACENONE, according to the following descriptions. Then press the Enter key.

- 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, note that it is not always possible to cancel a command.
- Enter ABEND to abnormally terminate QMF and produce an abend dump (if a DSQUDUMP 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.

Using error log reports from the Q.ERROR_LOG table

The Q.ERROR_LOG table is a QMF control table that logs information about resource problems and problems caused by possible software defects. The structure of the table is shown in Table 61.

Table 61. Structure of the Q.ERROR_LOG table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
DATESTAMP	CHAR	8	no	The date on which the error occurred. It is in the form yyyymmdd.
TIMESTAMP	CHAR	5	no	The time at which the error occurred. It is in the form hh:mm, where hh is the hour and mm is the minute.
USERID	VARCHAR	128	no	The logon ID or, in CICS, the terminal ID of the user who experienced the error.
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. You need to know the terminal ID of the user who experienced the problem and the approximate time the problem occurred. Figure 125 on page 350 shows the format of the query.

```
SELECT TIMESTAMP, MSG_NO, MSGTEXT
  FROM Q.ERROR_LOG
  WHERE USERID = 'terminal_id'  (for CICS)
  WHERE USERID = 'user_id'      (for other than CICS)
    AND DATESTAMP = 'date'
    AND TIMESTAMP BETWEEN 'time1' AND 'time2'
ORDER BY TIMESTAMP, MSG_NO, MSGTEXT
```

Figure 125. Querying the error log for problem information

Be sure to use valid formats for the date and times you supply.

Reporting a problem to IBM

Before you report a problem to IBM, check IBM's Software Support Facility (SSF) to see if the problem has already been reported. For unreported problems, IBM support center representatives prepare an Authorized Program Analysis Report (APAR), which includes useful information about how to solve the problem.

If you have access to the SSF through *ServiceLink* or some other facility, read "Using ServiceLink to search for previously reported problems" for instructions on how to develop a string of search keywords that help you find the problem. If you do not have access to ServiceLink, you can go directly to "Working with your IBM support center" on page 353.

Using ServiceLink to search for previously reported problems

Search the SSF by constructing a string of search words that describe your problem. Every string of search words for QMF begins with the component ID 566872101 and a release number (shown in Table 62) that matches the QMF national language environment in which you experienced the problem.

Table 62. Release numbers for QMF base product and NLFs

NLF	ID
Brazilian Portuguese	65A
Danish	654
English	610
French	655
German	656
Italian	657
Japanese	658
Korean	659
Spanish	65B

Table 62. Release numbers for QMF base product and NLFs (continued)

NLF	ID
Swedish	65C
Swiss French	65D
Swiss German	65E
Uppercase English	651

The flowchart in Figure 126 on page 352 shows how to develop your search words as you determine each characteristic of the problem.

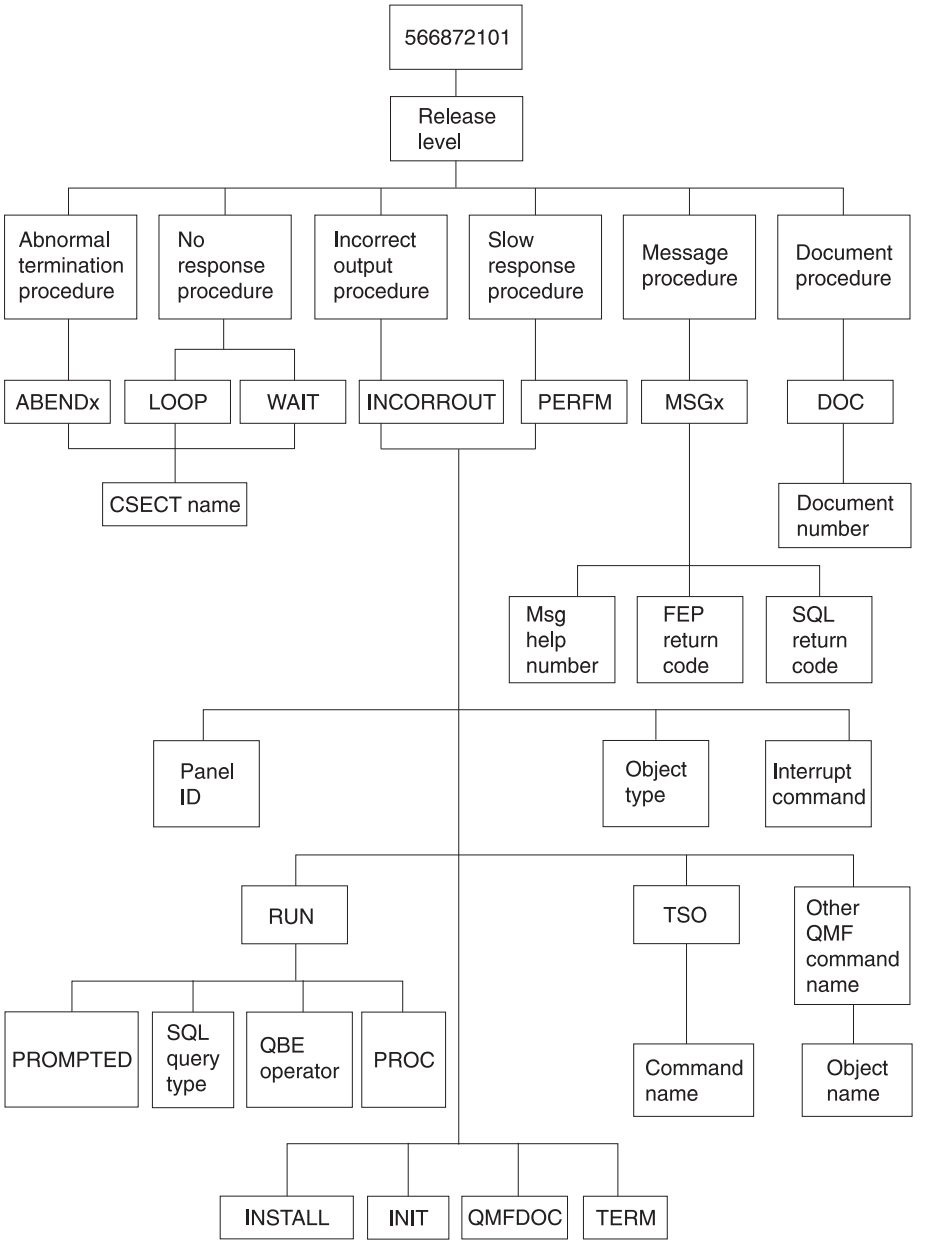


Figure 126. Chart of keyword types. Move from the top to the bottom of this chart to determine your keywords.

For example, if the problem you are searching for is an abend type of 0C4 that occurred in the DSQFDTBL control section (CSECT) when a user was running an English QMF session, use this search phrase:

```
566872101 09 ABEND0C4 DSQFDTBL
```

To find the CSECT name, look in the section of the trace output that has the heading ABEND CSECT NAME. The CSECT name is set off by asterisks. See "Using the QMF trace facility" on page 338 for more information on how to use the QMF trace facility.

For more information on searching the SSF for known QMF problems, see the *ServiceLink User's Guide*.

Working with your IBM support center

If you are having trouble diagnosing the problem and have used the diagnosis aids explained in this chapter, contact your IBM Support Center to report the problem.

To help diagnose the problem, your support center 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, which is explained in "Using the QMF trace facility" on page 338. The IBM representative might also need documentation produced by other diagnosis aids shown in Table 59 on page 336. This documentation can help the representative recreate the problem.

Part 3. Appendixes

Appendix A. Miscellaneous

What if it did not work?

During the installation process, you will receive some informational messages that you can safely ignore; others are warning or error messages that require corrective action. This section describes some of the most common errors that occur during installation. This list is not meant to replace the Messages and Codes manuals for QMF or other products. If you do not find the message on this list, consult the appropriate *Messages and Codes* manual.

Error messages you might see

You may receive one or more of the following error messages.

ABENDASRA

- On QMF startup:
  - Make sure the GDDM link-edit ran successfully.
  - Make sure the GDDM IVPs are successful in the region base.
  - Make sure QMF correctly link-edits.
  - Make sure the region allocates the QMF Version 8.1 LOADLIBs and map groups.
- In module DSQQMFE CSECT ADM

  The problem is probably a GDDM failure. Verify that GDDM is correctly installed and tailored for CICS. Verify that GDDM is located in the same CSI zone as CICS.
- In module DSQQMFE CSECT DSQEGINT

  Verify that GDDM is customized for CICS and that the PPT entry exists for the GDDM module ADMASPLC.
- In module DSQQMFE CSECT DSQIELI

  Verify that the PPT entry exists for the DB2 UDB for z/OS interface module DSQIELI.
- In module DSQCBST CSECT DSQCMCVP

  After QMF Service has been applied, verify that a z/OS LLA REFRESH has been done in case QMF CODE is in the Lookaside Library.
- ABEND0C1 with FFFFFFFE in R15

  Rerun DSQ1ELNK, especially after applying QMF maintenance
- On exit of QMF

  Check that the governor is linked correctly. Review job DSQ1EGLK.
- With ABEND0C4 and DFHSM0102

This error occurs when running a query or when pressing the Help function key. Make sure that the FCT for DSQPNLE has RECFM=V.

- On issuing HELP or RUN commands

  The QMF data set DSQPNLE, 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.

  – Verify that the FCT entry is defined correctly.
  – Verify that a DD statement for DSQPNLE exists in the job stream that starts the CICS region. DD statements are described in "Update CICS startup job stream" on page 38.

  Look for console error messages related to the DSQPNLE data set.

AEY9 ABEND

The DB2 UDB for z/OS attachment facility is not active in the CICS region. Start the attachment facility using the DSNC transaction.

AZTS ABEND

Make sure that GDDM is running with IOSYNCH=YES.

DSNT302I

Invalid name `profilex`. This is a normal message produced by DSQ1TBJ2; ignore the message.

DSQ10297

Invalid subsystem ID. This error can occur on ISPF startup or when using the callable interface. Check your ISPF startup parameters to make sure that s=xxxx or DSQSSUBS=xxxx. See "Starting QMF with ISPF" on page 31 for more details.

DSQ10493

This message indicates a database authorization error. Verify that the DB2 UDB 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 an entry of:

```
DSNCRCT TYPE=ENTRY,TXID=QMFE,PLAN=QMF810,AUTH=DEPT1
```

In this example, the authorization ID is DEPT1, and the plan ID is QMF810.

DSQ36805

SQLCODE 805. This error occurs during startup. Record all the tokens returned from the SQLCODE 805, and follow the directions in *DB2 UDB for z/OS Message and Codes* for the -805.

DSQI004I

GDDM error.

DSQI0026

This message usually occurs on startup. Make sure that the QMFE transaction is entered from a clear screen.

G050 ABEND

Verify that the release level of GDDM that you tailored for CICS matches the release level of GDDM that you are using in the job stream to start the CICS region.

IDC3012I

Entry QMFCAT.DSNDBC.DSQDBCTL.PROFILEX.I0001.A001.

IDC3009I

**VSAM catalog return code is 8 - reason code is IGGOCLAS3-42.

IDC0551I

**Entry QMFCAT.DSNDBC.DSQDBCTL.PROFILEX.I001 A001 not deleted.

These are normal messages that occur during the delete and purge of the VSAM cluster, when running DSQ1VSTP; ignore the messages.

IEW0342

Library does not contain module xxxxxxxx.

QMF is attempting to replace a module that does not yet exist. You receive this message for each load module link-edited.

IEW0461

You received this warning message because of one of the following reasons:

- The symbol printed is an unresolved external reference.
- NCAL is specified.
- The reference is marked for restricted NO=CALL or NEVERCALL.

This message occurs for three load modules (DSQUXIA, DSQUXIC, and DSQUXIP). These modules are the sample assembler, COBOL, and PL/I user exits; ignore these messages.

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.

The QMF IVP includes a step to print a query, which requires a GDDM nickname. If you use GDDM nicknames at your installation, change the PRINT QUERY statement in the IVP procedure to PRINT QUERY (PRINTER = *gddmnickname*. The procedure for creating GDDM printer nicknames is

discussed in Chapter 15, "Enabling users to print objects," on page 181. If you do not use GDDM nicknames at your installation, replace the PRINT in the IVP procedure with PRINT PROFILE. QMF prints the profile without using nicknames.

Warning messages

Warning messages after you start QMF might be caused by:

- Same AUTHID as TSO

  If you use the same database AUTHID in TSO and CICS, you can use a QMF command synonym table that contains TSO commands. Although this warning does not affect running QMF, such command synonyms are not available during the CICS session.

  To allocate a unique profile for the CICS session and eliminate the warning message, see the discussion in "Tailor the QMF profile" on page 38.

- Other factors

  When a warning message is issued, the cause of the warning is written to the QMF trace data set, DSQDEBUG. The ddname DSQDEBUG is described in the job stream that started the CICS region.

What if I did not get an error message?

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 8 Release 1, but instead points to another release. In this case, make sure your ADMGGMAP ddname points to the QMF810.DSQMAPn data set. For further details on troubleshooting in general and incorrect output specifically, see Chapter 21, "Troubleshooting and problem diagnosis," on page 329.

Access to QMF trace data set DSQDEBUG

If you have a warning or system error during QMF initialization, you must look at the QMF trace data set to understand the reason for the error. In CICS, the trace data set is described as an extra partition data set. The trace data set is described in the CICS tables by a DCT TYPE=SDSCI and a DCT TYPE=EXTRA, as shown in Figure 127 on page 361.

```
TITLE 'DSQDCTSD - QMF SDSCI ENTRIES'
* TRACE DATA SET
      DFHDCT TYPE=SDSCI,DSCNAME=DSQDEBUG,
            RECFORM=VARBLK,
            RECSIZE=121,
            BLKSIZE=6050,
            TYPEFILE=OUTPUT
*
      TITLE 'DSQDCT - CICS DESTINATION CONTROL TABLE'
*
* TRACE DATA SET
*
DSQD DFHDCT TYPE=EXTRA,DESTID=DSQD,DSCNAME=DSQDEBUG,RSL=1
```

Figure 127. 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 that recorded it.

To look at the trace data set while the CICS region is active, you must close the trace data set using the CICS queue ID DSQD. You can do this using the CICS-supplied transaction CEMT. 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 CICS-supplied transaction CEMT to open the trace data set using the CICS queue ID DSQD.

DB2 QMF Version 8.1 product limitations in CICS on z/OS

Some functions provided by QMF are dependent on underlying system services and other program products that are available in z/OS and TSO, but not in CICS on z/OS. ISPF is not available in CICS. REXX is not available in CICS. The following QMF functions or programs are not supported in QMF when running in CICS; these functions depend on ISPF (as well as other services in some cases):

- Report calculations
- Conditional formatting
- Column definition
- Procedures with logic

Other products are not available in CICS:

- Repository Manager
- Document Interface

The EDIT PROC and EDIT QUERY commands are not available in CICS. However, it is possible to edit procedures and queries using the DISPLAY command with QMF:

- TSO command
- CONNECT command (when issued to connect to another database)
- Remote unit of work and distributed unit of work
- QMF client/server components

Appendix B. QMF objects residing in DB2

The following tables show a DBA the QMF objects that reside in the database. The tables are intended to summarize all the database objects that are needed to run DB2 QMF Version 8.1 in the DB2 subsystem. These tables are not intended as replacements for the Installation jobs outlined in this book, but merely as a guide if database object recovery is needed.

QMF plans

Table 63 describes the plans shipped with DB2 QMF for TSO/CICS.

Table 63. QMF plans

Plan name	Bind job	Notes
QMF810	DSQ1BINR	General QMF plan
DSQIN810	DSQ1BSQL	QMF plan used for installation jobs only

QMF packages

Table 64 describes the package shipped with QMF.

Table 64. QMF Packages

Package name	Bind job
DSQE*	DSQ1BPKG JCL (z/OS)
To a remote server: DSQE*	DSQ1BPKG JCL (z/OS any supported server)

QMF control tables and table spaces for TSO/CICS

Table 65 on page 364 shows the control tables shipped with QMF.

Note: iSeries requires a Collection ″Q″ be created before these QMF DB storage structures can be created. There are no nodegroups, table spaces or dbspaces in iSeries.

Table 65. QMF Objects, Control Tables, Save Data Tables, and Sample Tables

Control table name	Table space	Table space size (in 1K 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, and procedures in the database.	Q.OBJECT_ DIRECTORYX
Q.OBJECT_DATA	DSQTSCT3	5000 primary, 200 secondary	Contains queries, forms, and procedures represented in an internal QMF format.	Q.OBJECT_ OBJDATAX
Q.OBJECT_ REMARKS	DSQTSCT2	200 primary, 20 secondary	Contains comments that were saved when queries, forms, and procedures were created or replaced.	Q.OBJECT_ REMARKSX
Q.COMMAND_ SYNONYMS	DSQTSSYN	100 primary, 20 secondary	Contains information on the command synonyms.	Q.COMMAND_ SYNONYMNSX
Q.RESOURCE_ TABLE	DSQTSGOV	100 primary, 20 secondary	Contains resource control information passed to the governor exit routine.	Q.RESOURCE_ INDEX

Table 65. QMF Objects, Control Tables, Save Data Tables, and Sample Tables (continued)

Control table name	Table space	Table space size (in 1K units)	Table content	Index
Q.ERROR_LOG	DSQTSLOG	100 primary, 20 secondary	Contains information on system, resource, and ″unexpected condition″ errors. This information is more detailed than that found in error messages.	none
Q.DSQ. RESERVED	DSQTSRDO	12 primary, 4 secondary	Contains information on used by QMF during installation. IMPORTANT: DO NOT MODIFY THIS TABLE	none

QMF views

The following table describes the views shipped with QMF.

Table 66. Views shipped with QMF

View name	Table viewed	Operating system
Q.DSQEC_ALIASESL	SYSIBM.SYSTABLES	z/OS
	SYSCAT.TABLES	workstation
	QSYS2.SYSTABLES	iSeries
Q.DSQEC_COLS_LDB2L	SYSIBM.SYSCOLUMNS	z/OS
	SYSIBM.SYSTABAUTH	z/OS
	SYSCAT.COLUMNS	workstation
	SYSCAT.TABAUTH	workstation
	QSYS2.SYSCOLUMNS	iSeries
Q.DSQEC_COLS_RDB2L	SYSIBM.SYSCOLUMNS	z/OS
	SYSIBM.SYSTABAUTH	z/OS
Q.DSQEC_QMFOBJSL	Q.OBJECT-DIRECTORY	All operating systems
	Q.OBJECT_REMARKS	All operating systems

Table 66. Views shipped with QMF (continued)

View name	Table viewed	Operating system
Q.DSQEC_TABS_LDB2L	SYSIBM.SYSTABAUTH	z/OS
	SYSIBM.SYSTABLES	z/OS
	SYSCAT.TABAUTH	workstation
	SYSCAT.TABLES	workstation
	QSYS2.SYSTABLES	iSeries
Q.DSQEC_TABS_RDB2L	SYSIBM.SYSTABAUTH	z/OS
	SYSIBM.SYSTABLES	z/OS
Q.RESOURCE_VIEW	Q.RESOURCE_TABLE	All operating systems
Q,DSQ_RESERVED_DB	SYSIBM.SYSCOLUMNS	z/OS
	QSYS2.SYSCOLUMNS	iSeries
	SYSCAT.COLUMNS	workstation
Q.DSQ_RESERVED_OBJ	Q.OBJECT_DIRECTORY	All operating systems

Several of these views are based on DB2 system tables and are used by QMF for the LIST and DESCRIBE functions.

You can create/recreate all QMF control table views on any supported DB2 database from z/OS by running job DSQ1BVW JCL. This job will DROP and CREATE all QMF control table views and GRANT necessary authorities.

On z/OS, if you want to enable QMF control table views for DB2 secondary authorization IDs, you must run this job to refresh the QMF views for that DB2 database.

VSAM clusters for TSO/CICS

Table 67 shows the VSAM clusters shipped with QMF.

Table 67. VSAM clusters

Cluster name	Object that cluster is needed for
QMFDSN.DSNDBC.DSQDBCTL.DSQTSCT1.I0001.A001	DSQTSCT1
QMFDSN.DSNDBC.DSQDBCTL.DSQTSCT2.I0001.A001	DSQTSCT2
QMFDSN.DSNDBC.DSQDBCTL.DSQTSCT3.I0001.A001	DSQTSCT3
QMFDSN.DSNDBC.DSQDBCTL.DSQTSPRO.I0001.A001	DSQTSPRO
QMFDSN.DSNDBC.DSQDBCTL.DSQTSLOG.I0001.A001	DSQTSLOG
QMFDSN.DSNDBC.DSQDBCTL.DSQTSGOV.I0001.A001	DSQTSGOV

Table 67. VSAM clusters (continued)

Cluster name	Object that cluster is needed for
QMFDSN.DSNDBC.DSQDBCTL.DSQTSSYN.I0001.A001	DSQTSSYN
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_OBJDATAX
QMFDSN.DSNDBC.DSQDBCTL.PROFILEX.I0001.A001	Q.PROFILEX
QMFDSN.DSNDBC.DSQDBCTL.COMMANDR.I0001.A001	Q.COMMAND_SYNONYMSX

QMF sample tables for TSO/CICS

Table 68 describes the sample tables.

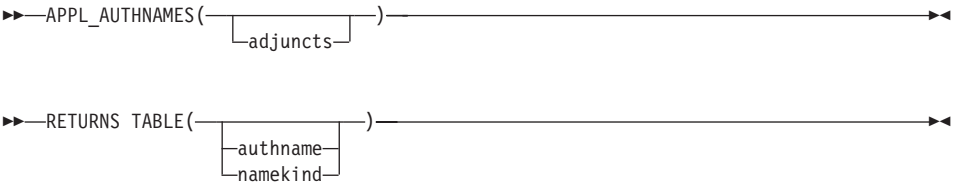
Table 68. Sample tables

Table	Contains information about:
Q.ORG	The company organization
Q.STAFF	The company personnel
Q.APPLICANT	New candidates for hire
Q.PRODUCTS	The company's products
Q.SALES	Sales and commissions
Q.PROJECT	Projects undertaken, by department
Q.INTERVIEW	Interviews of new hires
Q.SUPPLIER	Vendor information
Q.PARTS	Product parts data

Appendix C. QMF user-defined functions

APPL_AUTHNAMES

The syntax description for the user defined function table is:



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.

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:

```
'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:

- **authname** CHARACTER (8)

  The name for an authorization ID of the current process.
- **namekind** CHARACTER(1)

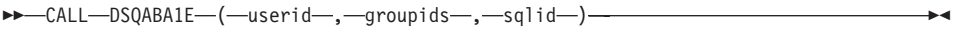
  A classification code for the name value in AUTHNAME:
  - **1** Primary authorization ID or user name
  - **2** Secondary authorization ID or group name
  - **3** Current authorization ID

    This applies only when the CURRENT SQLID is neither the primary ID nor a secondary ID of the current process.
  - **9** Adjunct name value

This applies only when the ADJUNCT parameter is used and the identifiers it specifies are not authorization IDs of the current process.

CALL DSQAB1E

The syntax description for the stored procedure interface is:



The DSQAB1E stored procedure returns the DB2 authorization IDs for the currently running process. The schema name is Q.

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 into 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 syntax description for the diagnostic user defined function is:

The DSQABA1E function returns diagnostic information that can assist IBM Service with problem diagnosis. The schema name is Q.

The result of the function is a character string with a datatype of VARCHAR and an actual length not greater than 5,300 bytes. This string is suitable for formatting in a QMF report with column setting of WIDTH = 53 and an EDIT code of CW.

Appendix D. How QMF and GDDM programs are defined to CICS

QMF for TSO/CICS provides the jobs necessary to define QMF programs to CICS and load GDDM definitions and chart formats for QMF panels. Use this section if you need to know how QMF programs are defined and how GDDM definitions are loaded during QMF installation.

How QMF programs are defined to CICS

During QMF installation, the default transaction ID QMF*n* is defined for QMF, where *n* is a national language identifier from Table 1 on page ix. The transaction ID is defined in either the CICS program control table (PCT) or the system definition (CSD) file.

Resident QMF programs

During QMF installation, the following programs are defined as resident in CICS:

DSQQMF
DSQQMFn
DSQCBST
DSQCnLTT
DSQCnBLT
DSQUEGV3
DSQUECIC

CICS treats programs with RMODE(ANY) 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. See "How nonresident GDDM programs affect QMF" on page 372 for more information.

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 program. However, if the needs of your installation 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, or specify RES=NO on the DFHPPT TYPE=ENTRY macro to change the value in the program processing table (PPT).

How GDDM definitions are loaded during QMF installation

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 QMF 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 tailor GDDM for CICS, consider making the GDDM programs resident, because certain GDDM programs are loaded when QMF is started whether or not you use QMF's charting functions. See the *CICS/MVS Performance Guide* for more information on how to decide which programs should be resident. For more information on tailoring GDDM for CICS, see *GDDM System Customization and Administration*.

How chart formats are defined

The QMF default installation stores chart formats, chart data, and GDF data in the GDDM file ADMF. 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 function after QMF installation

If you install GDDM-PGF after you install QMF, you need to fully install and tailor GDDM-PGF for CICS, rather than merely restoring the product to a sublibrary.

If you use GDDM 3.1, you need to install GDDM-PGF 2.1.2.

After you install GDDM-PGF and tailor it, you can verify the installation by running the CICS ADMC transaction, which is predefined by GDDM during GDDM tailoring for CICS. No further customization of the chart formats is necessary; these formats were defined for you during QMF installation.

Using transaction routing to control resource use

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. Both methods are described in the *CICS/OS390 Intercommunication Guide*.

Appendix E. Migration and fallback between QMF releases

Note: Skip this section if you are installing QMF for the first time.

After QMF Version 8.1 has been successfully installed, check to see if your users are still using the earlier release of QMF. You need to help them operate the new release; you might need to:

- Grant them access to the DB2 QMF Version 8.1 application plan
- Provide them with an appropriate QMF profile
- Make objects created earlier (queries and forms, for example) available for QMF sessions under the new release.

What is meant by migration?

Migration is the process of carrying out the steps in the Installation section of this book. In QMF Version 8.1, there are two possible migration scenarios:

- A Compatibility mode migration from a previous release of QMF (versions 3.3, 6.1, 7.1, or 7.2) to version 8.1
- A migration from QMF Version 8.1 Compatiblity mode to QMF Version 8.1 New Function mode

This appendix assumes that QMF Version 8.1 was installed according to the instructions in this book. If it was not, or if some of the settings have been changed, parts of the discussion might not apply.

Multiple releases of QMF

Prior to QMF Version 8.1, multiple releases of QMF could access one DB2 UDB database, and all releases used the same QMF control tables and QMF objects. With QMF Version 8.1, only QMF Version 8.1 installed in QMF Compatibility mode can access the same DB2 UDB database as previous QMF releases.

If QMF Version 8.1 New Function mode is installed into a DB2 UDB database, no previous releases of QMF can access that database.

Note: A QMF Version 8.1 New Function mode installation can connect to a QMF Version 8.1 Compatibility mode installation and conversely, a QMF Version 8.1 Compatibility mode installation can connect to a QMF Version 8.1 New Function mode installation.

DB2 subsystems and migration

When you migrate users, the new and old versions of QMF might be on the same DB2 subsystem or on two different subsystems.

- If the two releases of QMF are on the same DB2 subsystem, read "Migrating QMF on the same DB2 UDB subsystem."
- If the two releases of QMF are not on the same DB2 subsystem, read "Migrating QMF across different DB2 subsystems."

Migrating QMF on the same DB2 UDB subsystem

Read this section to migrate when both releases of QMF are on the same DB2 UDB subsystem.

Note: Only a QMF Version 8.1 Compatibility mode installation can coexist in the same DB2 UDB subsystem with a previous release of QMF. This section refers only to QMF Version 8.l Compatibility mode migrations.

Providing a QMF profile on z/OS

At the start of a QMF session, a user's QMF profile comes from some row of the Q.PROFILES table. With both releases of QMF in the same DB2 subsystem, the two releases use the same Q.PROFILES table.

If a user has a primary authorization ID different from the TSO logon ID, the DSQSPRID parameter should have a value of TSO ID when you start QMF. Otherwise, insert a user row in Q.PROFILES with CREATOR set to the primary authorization ID.

For QMF Version 3.3, 6.1, 7.1, and 7.2 Users: There are no new columns in the Q.PROFILES table if you are migrating from Version 3.3 or higher.

Making objects from the earlier release available under DB2 QMF Version 8.1

If both releases of QMF are on the same DB2 subsystem, all the DB2 objects (tables and views, for example), are available under DB2 QMF Version 8.1 (both Compatibility mode and New Function mode) if they are available under the earlier release. All the queries, forms, and procedures are also available, but some might be unusable under QMF Version 8.1. This topic is discussed in "Migrating QMF objects" on page 379.

Migrating QMF across different DB2 subsystems

This section describes how to migrate when both releases of QMF are in different DB2 subsystems.

When the DB2 subsystems are different, migration is complicated by the fact that QMF objects in the database for the earlier QMF release are not available to Version 8.1 users. Nor are these objects in the DB2 QMF Version 8.1 database available to users of the earlier QMF release.

The tables and views required by QMF must be made available in the new subsystem.

Providing a QMF profile

The installation process creates a new Q.PROFILES table when QMF Version 8.1 is in a different DB2 subsystem.

This newly created table contains a single SYSTEM row. The values assigned to the columns appear in Table 69

Table 69. Installation-supplied SYSTEM row values

Column	Value
CREATOR	SYSTEM
CASE	UPPER
DECOPT	PERIOD
CONFIRM	YES
WIDTH	132
LENGTH	60
LANGUAGE	SQL
SPACE	DSQDBDEF.DSQTSDEF
TRACE	NONE
PRINTER	blank
TRANSLATION	ENGLISH
PFKEYS	Zero-length string
SYNONYMS	Q.COMMAND__SYNONYMS
RESOURCE__GROUP	SYSTEM
MODEL	REL
ENVIRONMENT	Null

If CICS is installed, there is an additional SYSTEM row, in which SYNONYMS is set to null and ENVIRONMENT is set to CICS.

With only the SYSTEM row in the table, users begin their Version 8.1 sessions with the QMF profile provided by this row. This profile can differ from profiles on earlier QMF releases. You can recreate the earlier profiles with a series of INSERT queries, but users can also do this for themselves with SET or SAVE PROFILE.

The PFKEYS, SYNONYMS, and RESOURCE__GROUP parameters play key roles in customizing the QMF environment. For a brief description of each, see Table 69 on page 377

Users cannot, however, change the values of the PFKEYS, SYNONYMS, and RESOURCE__GROUP parameters with SET or SAVE PROFILE. You must do this with an UPDATE query on the Q.PROFILES table. For an example of this, see "Activating new function key definitions" on page 224.

Making objects from the earlier release available under QMF Version 8.1

This section applies to QMF Compatibility mode and New Function mode installations.

DB2 tables and QMF objects can be exported from a subsystem under an earlier QMF release and then imported under QMF Version 8.1.

To migrate DB2 tables, any user with the proper DB2 authority can:

1. Unload the tables using a DB2-supplied application program, DSNTIAUL. For more on this program, see the *DB2 UDB for z/OS Administration Guide*.
2. Load the unloaded tables into the Version 8.1 DB2 subsystem using the DB2 loader.

If the two versions of QMF are on different z/OS systems, use the available networking facilities to send the exported objects and unloaded tables to the system containing QMF Version 8.1.

To migrate QMF queries, forms, procedures, and applications, make sure you read the following section, "Migrating QMF objects" on page 379.

If you have DB2 QMF High Performance Option (HPO) installed, you can use the HPO Object Manager for assistance with the migration of QMF objects from one DB2 subsystem to another.

For more information on Data Refresher or the DB2 QMF High Performance Option, see the web site at: http://www.ibm.com/software/data/qmf.

Views and synonyms

If you use QMF to export tables from a database and import them to a different database, you must create any views, indexes, synonyms, and authorizations on that table at the new database.

Migrating QMF objects

This section describes migration considerations for QMF objects. Most objects created under earlier releases of QMF can be used under DB2 QMF Version 8.1. (For information about migrating back to an earlier release of QMF, see "Fallback" on page 381.)

Queries and forms

All queries and forms created under earlier releases of QMF can be used under DB2 QMF Version 8.1.

Procedures

Procedure objects that were saved or exported in Version 3.3 can be displayed or imported under Version 8.1, and they can be run if the abbreviations for commands and options (if any are used) are valid in Version 8.1. Version 3.3 procedures containing commands or applications requiring ISPF run only if DB2 QMF Version 8.1 is started as an ISPF dialog. Procedures written in English and saved or exported in QMF Version 3.3 can be imported and run without modification in a Version 8.1 NLF session (a QMF session where English is not the presiding language) if the command language global variable is set to accept English commands.

Some procedures from earlier releases will not work properly if they issue commands with verbs that are also verbs for installation-defined commands. To ensure this does not happen under DB2 QMF Version 8.1, users can add `QMF` before all commands. This identifies these commands as standard QMF commands instead of installation-defined commands. It lets these procedures run under DB2 QMF Version 8.1. For more information on installation-defined commands, read Chapter 16, "Customizing QMF commands," on page 197.

Migrating applications

Previous QMF release applications containing commands requiring ISPF run only if Version 8.1 is started as an ISPF dialog. Applications that issue commands written in English and that run with previous QMF releases can be run without modification in a Version 8.1 NLF session (a QMF session where English is not the presiding language) if the command language global variable has been set to accept English commands.

Callable interface considerations

If you want to use the LIBDEF function in your QMF applications that were link edited prior to DB2 QMF Version 8.1 and that use the callable interface, you must re-link edit your application using the DB2 QMF Version 8.1 interface module.

Other migration considerations

This section describes other migration considerations for QMF, including special considerations for the environment that is being used for QMF.

Governor in CICS

If you plan to use the IBM-supplied governor, replace it with the new version. The governor continues to function as before, and its contents are unchanged.

User edit routine in TSO and native z/OS batch

For QMF Version 8.1, you need to relink your user edit code. For more information about relinking your user edit code, see Chapter 18, "Creating your own edit codes for QMF forms," on page 227.

Callable interface in CICS

If you are migrating from QMF Version 3.3 or later, the interface between the QMF-supplied function call and the main QMF program has changed from a CALL interface to an EXEC CICS LINK interface. The new interface provides better isolation from the user program and the QMF product. Because the interface has changed, it is necessary to relink-edit your programs that use the callable interface.

Export/import support for CICS on z/OS

When running QMF in CICS, you must set the execution key of the QMF module DSQCBST to CICS (EXECkey=CICS) if you plan to use the QMF EXPORT or IMPORT commands and CICS storage protection (SIT STGPROT=YES) is being used. This avoids abnormal terminations (ABENDASRA or ABEND0C4) in IGG0191I conditions.

Migration considerations and support

QMF provides a migration capability that allows you to choose between the recommended use of CICS temporary storage or transient data queues and the volatile use of TSO data sets. After QMF Version 8.1 is installed, the default use of CICS temporary storage and transient data queues, is active. If you do not want to use the TSO data sets, there are no migration considerations.

If you do want to use the TSO data sets, then you must disable the QMF export/import control module, DSQCTLXI. To do this, use the CICS-supplied CEMT transaction. For example:

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

DSQCTLXI can also be disabled by removing it from the CICS CSD or PCT table. After you disable DSQCTLXI, all QMF sessions running in CICS use the TSO data set support for export and import commands.

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

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

Migrating from version 3, 6, or 7

In QMF Version 8.1, the QMF installation CLIST, DSQ1EINS, and the associated ISPF installation panels no longer exist. QMF must be installed through batch jobs. The QMF installation creates three new SMP/E target libraries that were formally non-SMP/E managed data sets.

Table 70. Former non-SMP/E managed data sets and their replacement SMP/E target libraries

Former non-SMP/E managed data sets	New SMP/E target libraries
QMF.DSQMAPn	QMF.SDSQMAPn
QMF.DSQCHART	QMF.SDSQCHRT
QMF.DSQPVARn	QMF.SDSQPVRn

For a complete list of QMF Version 8.1 views and database objects, see Appendix B, “QMF objects residing in DB2,” on page 363

Fallback

Fallback is the process of migrating a user back to the earlier release of QMF. Cleanup is the process of removing the earlier release from z/OS. Cleanup is described in “Clean up after installation” on page 58 and is not discussed here.

Fallback is not necessary unless the two versions of QMF are running from the same DB2 subsystem.

What do we mean by fallback?

Fallback is the process of migrating users from QMF Version 8.1 Compatibility mode to an earlier release. There is no fallback mode from QMF Version 8.1 Compatibility mode to QMF Version 8.1 New Function mode. Cleanup is described in “Clean up after installation” on page 58 and is not discussed here.

Re-establishing the earlier profiles

If logon IDs are different from primary authorization IDs and the CREATOR values were updated to use the primary authorization IDs, then they must be restored to the logon IDs as part of fallback

Using DB2 QMF Version 8.1 objects under earlier releases

This is largely preventive. While there is still a chance for fallback, make certain that your users understand the compatibility rules given previously in this appendix. If you have not looked at these rules already, read "Migrating QMF objects" on page 379.

If you fall back to the earlier QMF release, some objects created under QMF Version 8.1 cannot be used in the earlier environment. Consider this when planning for a possible fallback. The following list contains the restrictions that apply when you use some Version 8.1 objects in earlier releases.

- Data objects

  QMF data objects exported using the QMF dataformat cannot be imported into earlier releases of QMF. IXF data objects exported using IXF dataformat can be imported into earlier releases of QMF provided that they do not use column names which are longer than 18 characters.

- Queries

  Some restrictions apply to Version 8.1 queries for fallback to earlier releases:

  – Prompted queries: You can display and import Version 8.1 prompted queries in earlier releases provided they do not contain variables, or expressions with more than the old 55 or 65 character limit. Prompted queried that are exported or saved while running Version 8.1 in New Function mode cannot be used in earlier versions of QMF.

- Procedures

  Procedure objects exported from Version 8.1 can be imported into earlier releases, and they can be run if the new QMF commands or command syntax are not used. Procedure objects saved with Version 8.1 cannot be displayed with earlier releases unless you first export them from Version 8.1 and import them into the earlier release.

- Procedures or applications containing QMF commands that cannot be run under the earlier release

  These commands might fail to run for a number of reasons. See "Using QMF Version 8.1 commands under earlier releases" for details.

For specific differences between an earlier QMF release and DB2 QMF Version 8.1, compare the two releases of *DB2 QMF Reference*.

Using QMF Version 8.1 commands under earlier releases

QMF Version 8.1 procedures and applications might run incorrectly under an earlier QMF release because they contain commands that the earlier release cannot run. Some commands:

- Do not exist in the earlier release.
- Contain options that operate differently in the earlier release. For example, the DRAW command has the same syntax as before, but now produces

different results. All keywords now have double quotes; therefore, the users no longer have to add the quotes, and any tools used to provide double quotes are no longer necessary.

Appendix F. Notices

This information was developed for products and services offered in the U.S.A. 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
Armonk, NY 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
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 states 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.

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 Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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 information 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
C/370
CICS
COBOL/370
DataJoiner
DB2
DB2 Information Integrator
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
GDDM
IBM
IBMLink
IMS
iSeries
MVS
OS/390
Parallel Sysplex
PL/I
QMF
RACF
S/390
SQL/DS
VM/ESA
VSE/ESA
VTAM
WebSphere
z/OS
zSeries

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other counrtries, 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.

Appendix G. Glossary of Terms and Acronyms

This glossary defines terms as they are used throughout the QMF library. If you do not find the term you are looking for, refer to the index in this book, or to the *IBM Dictionary of Computing*.

abend. The abnormal termination of a task.

ABENDx. The keyword for an abend problem.

aggregation 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 either 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. In DB2 UDB for OS/390, an alternate name that can be used in SQL statements to refer to a table or view in the same or a remote DB2 UDB for OS/390 subsystem. In OS/2, an alternate name used to identify a object, a database, or a network resource such as an LU. In QMF, a locally defined name used to access a QMF table or view stored on a local or remote DB2 UDB for OS/390 subsystem.

APAR. Authorized Program Analysis Report.

application. A program written by QMF users that extends the capabilities of QMF without modifying the QMF licensed program. Started from a QMF session by issuing a RUN command for a QMF procedure, an installation-defined command, or a TSO command that invokes a CLIST.

application requester. (1) A facility that accepts a database request from an application process and passes it to an application server. (2) In DRDA, the source of a request to a remote relational database management system.

The application requester is the DBMS code that handles the QMF end of the distributed connection. The local DB2 UDB for OS/390 subsystem to which QMF attaches is known as the application requester for QMF, because DB2 UDB for OS/390's application requester is installed within the local database manager. Therefore, an entire DB2 UDB for OS/390 subsystem (including data) is associated with the application requester, but the SQL statements are processed at the current location. This subsystem is called the "local DB2 UDB for OS/390".

With DB2 for VM and VSE the application requester runs in the same virtual machine as QMF; that is, no database is inherently associated with the DB2 for VM and VSE application requester.

application server. The target of a request from an application requester. (1) The local or remote database manager to which the application process is connected. The application server executes at the system containing the desired data. (2) In DRDA, the target of a request from an application requester. With DB2 UDB for OS/390, the application server is part of a full DB2 UDB for OS/390 subsystem.

With DB2 for VM and VSE, the application server is part of a DB2 for VM and VSE database machine.

application-support command. A QMF command that can be used within an application program to exchange information between the application program and QMF. These commands include INTERACT, MESSAGE, STATE, and QMF.

area separator. The barrier that separates the fixed area of a displayed report from the remainder of the report.

argument. An independent variable.

base QMF environment. The English-language environment of QMF, established when QMF is installed. Any other language environment is established after installation.

batch QMF session. A QMF session running in the background. Begins when a specified QMF procedure is invoked and ends when the procedure ends. During a background QMF session, no user interaction and panel display interaction are allowed.

bind. In DRDA, the process by which the SQL statements in an application program are made known to a database management system over application support protocol (and database support protocol) flows. During a bind, output from a precompiler or preprocessor is converted to a control structure called a package. In addition, access paths to the referenced data are selected and some authorization checking is performed. (Optionally in DB2 UDB for OS/390, the output may be an application plan.)

built-in function. Generic term for scalar function or column function. Can also be "function."

calculation variable. CALCid is a special variable for forms that contains a user-defined calculated value. CALCid is defined on the FORM.CALC panel.

callable interface. A programming interface that provides access to QMF services. An application can access these services even when the application is running outside of a QMF session. Contrast with command interface.

CICS. Customer Information Control System.

CMS. Conversational Monitor System.

column wrapping. Formatting values in a report so that they occupy several lines within a column. Often used when a column contains values whose length exceeds the column width.

command interface. An interface for running QMF commands. The QMF commands can only be issued from within an active QMF session. Contrast with callable interface.

command synonym. The verb or verb/object part of an installation-defined command. Users enter this for the command, followed by whatever other information is needed.

command synonym table. A table each of whose rows describes an installation-defined command. Each user can be assigned one of these tables.

commit. The process that makes a data change permanent. When a commit occurs, data locks are freed enabling other applications to reference the just-committed data. See also "rollback".

concatenation. The combination of two strings into a single string by appending the second to the first.

correlation name. An alias for a table name, specified in the FROM clause of a SELECT query. When concatenated with a column name, it identifies the table to which the column belongs.

CP. The Control Program for VM.

CSECT. Control section.

current location. The application server to which the QMF session is currently connected. Except for connection-type statements, such as CONNECT (which are handled by the application requester), this server processes all the SQL statements. When initializing QMF, the current location is indicated by the DSQSDBNM startup program parameter. (If that parameter is not specified, the local DB2 UDB for OS/390 subsystem

current object. An object in temporary storage currently displayed. Contrast with saved object.

Customer Information Control System (CICS). An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

database administrator. The person who controls the content of and access to a database.

database management system. A computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The database management system also has transaction management and data recovery facilities to protect data integrity.

database manager. A program used to create and maintain a database and to communicate with programs requiring access to the database.

database server. (1) In DRDA, the target of a request received from an application server (2) In OS/2, a workstations that provides database services for its local database to database clients.

date/time default formats. Date and time formats specified by a database manager installation option. They can be the EUR, ISO, JIS, USA, or LOC (LOCAL) formats.

date/time data. The data in a table column with a DATE, TIME, or TIMESTAMP data type.

DBCS. Double-byte character set.

DBMS. Database management system.

default form. The form created by QMF when a query is run. The default form is not created if a saved form is run with the query.

destination control table (DCT). In CICS, a table containing a definition for each transient data queue.

detail block text. The text in the body of the report associated with a particular row of data.

detail heading text. The text in the heading of a report. Whether or not headings will be printed is specified in FORM.DETAIL.

dialog panel. A panel that overlays part of a Prompted Query primary panel and extends the dialog that helps build a query.

distributed data. Data that is stored in more than one system in a network, and is available to remote users and application programs.

distributed database. A database that appears to users as a logical whole, locally accessible, but is comprised of databases in multiple locations.

distributed relational database. A distributed database where all data is stored according to the relational model.

Distributed Relational Database Architecture (DRDA). A connection protocol for distributed relational database processing that is used by IBM and vendor relational database products.

distributed unit of work. A method of accessing distributed relational data in which users or applications can, within a single unit of work, submit SQL statements to multiple relational database management systems, but no more than one RDBMS per SQL statement.

DB2 UDB for OS/390 introduced a limited form of distributed unit of work support in its V2R2 called system-directed access, which QMF supports.

double-byte character. An entity that requires two character bytes.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols that can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

EBCDIC. Extended Binary-Coded Decimal Interchange Code.

echo area. The part of the Prompted Query primary panel in which a prompted query is built.

EUR (European) format. A format that represents date and time values as follows:
- Date: dd.mm.yyyy
- Time: hh.mm.ss

extended syntax. QMF command syntax that is used by the QMF callable interface; this syntax defines variables that are stored in the storage acquired by the callable interface application and shared with QMF

gateway. A functional unit that connects two computer networks of different network architectures. A gateway connects networks or systems of different architectures, as opposed to a bridge, which connects networks or systems with the same or similar architectures.

global variable. A variable that, once set, can be used for an entire QMF session. A global variable can be used in a procedure, query, or form. Contrast with run-time variable.

Graphical Data Display Manager (GDDM). A group of routines that allows pictures to be defined and displayed procedurally through function routines that correspond to graphic primitives.

grouped row. A row of data in a QBE target or example table that is summarized either by a G. or a built-in function.

HTML. Hypertext Markup Language. A standardized markup language for documents displayed on the Internet.

ICU. Interactive Chart Utility.

INCORROUT. The keyword for incorrect output.

initialization program. A program that sets QMF program parameters. This program is specified by DSQSCMD in the callable interface. The default program for interactive QMF is DSQSCMD*n*, where *n* is the qualifier for the presiding language (';E'; for English).

invocation CLIST or EXEC. A program that invokes (starts) QMF.

ISO (International Standards Organization) format. A format that represents date and time values as follows:
- Date: yyyy-mm-dd
- Time: hh.mm.ss

ISPF. Interactive System Productivity Facility.

IXF. Integration Exchange Format: A protocol for transferring tabular data among various software products.

JCL. Job control language for OS/390.

job control. In VSE, a program called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to symbolic names, set switches for program use, log (or print) job control statements, and fetch the first phase of each job step.

JIS (Japanese Industrial Standard) format. A format that represents date and time values as follows:
- Date: yyyy-mm-dd
- Time: hh:mm:ss

keyword parameter. An element of a QMF command consisting of a keyword and an assigned value.

literal. In programming languages, a lexical unit that directly represents a value. A character string whose value is given by the characters themselves.

linear procedure. Any procedure *not* beginning with a REXX comment. A linear procedure can contain QMF commands, comments, blank lines, RUN commands, and substitution variables. 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.

local area network (LAN). (1) Two or more processors connected for local resource sharing (2) A network within a limited geographic area, such as a single office building, warehouse, or campus.

local data. Data that is maintained by the subsystem that is attempting to access the data. Contrast with remote data.

local DB2 UDB for OS/390. With DB2 UDB for OS/390, the application requester is part of a DB2 UDB for OS/390 subsystem that is running in the same MVS system as QMF. Therefore, an entire DB2 UDB for OS/390 subsystem (including data) is associated with the application requester, but the SQL statements are processed at the current location. This subsystem is where the QMF plan is bound.

When QMF runs in TSO, this subsystem is specified using DSQSSUBS startup program parameter. When QMF runs in CICS, this subsystem is identified in the Resource Control Table (RCT). The local DB2 UDB for OS/390 is the subsystem ID of the DB2 UDB for OS/390 that was started in the CICS region.

location. A specific relational database management system in a distributed relational database system. Each DB2 UDB for OS/390 subsystem is considered to be a location.

logical unit (LU). A port through which an end user accesses the SNA network to communicate with another end user and through which the end user accesses the functions provided by system services control points.

Logical Unit type 6.2 (LU 6.2). The SNA logical unit type that supports general communication between programs in a distributed processing environment.

MSGx. The keyword for a message problem.

MVS/ESA. Multiple Virtual Storage/Enterprise System Architecture (IBM operating system).

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

NLF. National Language Feature. Any of several optional features available with QMF that lets the user select a language other than US English.

NLS. National Language Support.

package. The control structure produced when the SQL statements in an application program are bound to a relational database management system. The database management system uses the control structure to process SQL statements encountered during statement execution.

panel. A particular arrangement of information, grouped together for presentation in a window. A panel can contain informational text, entry fields, options the user can choose from, or a mixture of these.

parameter. An element of a QMF command. This term is used generically in QMF documentation to reference a *keyword parameter* or a *positional parameter*.

partner logical unit. In SNA, the remote system in a session.

PERFM. The keyword for a performance problem.

permanent storage. The database where all tables and QMF objects are stored.

plan. A form of package where the SQL statements of several programs are collected together during bind to create a plan.

positional parameter. An element of a QMF command that must be placed in a certain position within the command.

primary panel. The main Prompted Query panel containing your query.

primary QMF session. An interactive session begun from outside QMF Within this session, other sessions can be started by using the INTERACT command.

procedure with logic. Any QMF procedure beginning with a REXX comment. In a procedure with logic, you can perform conditional logic, make calculations, build strings, and pass commands back to the host environment. See also "linear procedure."

profile. An object that contains information about the characteristics of the user's session. A stored profile is a profile that has been saved in permanent storage. A profile in temporary storage has the name PROFILE. There can be only one profile for each user.

Prompted Query. A query built in accordance with the user's responses to a set of dialog panels.

PSW. Program status word.

PTF. Program temporary fix.

QBE (Query-By-Example). A language used to write queries graphically. For more information see *Using QMF*

QMF administrative authority. At minimum, insert or delete priviledge for the Q.PROFILES control table.

QMF administrator. A QMF user with QMF administrative authority.

QMF command. Refers to any command that is part of the QMF language. Does **not** include installation-defined commands.

QMF session. All interactions between the user and QMF from the time the user invokes QMF until the EXIT command is issued.

qualifier. When referring to a QMF object, the part of the name that identifies the owner. 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'.

RDBMS. Relational database management system

relational database management system (RDBMS). A computer-based system for defining, creating, manipulating, controlling, managing, and using relational databases.

remote unit of work. (1) The form of SQL distributed processing where the application is on a system different from the relational database and a single application server services all remote unit of work requests within a single logical unit of work. (2) A unit of work that allows for the remote preparation and execution of SQL statements.

REXX. Restructured extended executor.

rollback. The process that removes uncommitted database changes made by one application or user. When a rollback occurs, locks are freed and the state of the resource being changed is returned to its state at the last commit, rollback, or initiation. See also *commit*.

row operator area. The leftmost column of a QBE target or example table.

run-time 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 run-time variable is only available in the current procedure or query. Contrast with global variable.

SBCS. Single-byte character set.

scalar. A value in a column or the value of a literal or an expression involving other scalars.

scalar function. An operation that produces a single value from another value and is expressed in the form of a function name followed by a list of arguments enclosed in parentheses.

SNAP dump. A dynamic dump of the contents of one or more storage areas that QMF generates during an abend.

SQLCA. Structured Query Language Communication Area.

SSF. Software Support Facility. An IBM online database that allows for storage and retrieval of information about all current APARs and PTFs.

Structured Query Language (SQL). A language used to communicate with DB2 UDB for OS/390 and DB2 for VSE or VM. Used to write queries in descriptive phrases.

subquery. A complete SQL query that appears in a WHERE or HAVING clause of another query (the main query or a higher-level subquery).

substitution variable. (1) A variable in a procedure or query whose value is specified either by a global variable or by a run-time variable. (2) A variable in a form whose value is specified by a global variable.

substring. The part of a string whose beginning and length are specified in the SUBSTR function.

System Log (SYSLOG). A data set or file in which job-related information, operational data, descriptions of unusual occurrences, commands, and messages to and from the operator may be stored.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

tabular data. The data in columns. The content and the form of the data is specified on FORM.MAIN and FORM.COLUMNS.

target table. An empty table in which example elements are used to combine columns, combine rows, or include constant values in a report.

temporary storage. An area where the query, form, procedure, profile, report, chart, and data objects in current use are stored. All but the data object can be displayed.

temporary storage queue. In CICS, a temporary storage area used for transfer of objects between QMF and an application or a system service.

thread. The DB2 UDB for OS/390 structure that describes an application's connection, traces its progress, provides resource function processing capability, and delimits its accessibility to DB2 UDB for OS/390 resources and services. Most DB2 UDB for OS/390 functions execute under a thread structure.

three-part name. A fully-qualified name of a table or view, consisting of a location name, owner ID, and object name. When supported by the application server (that is, DB2 UDB for OS/390), a three-part name can be used in an SQL statement to retrieve or update the specified table or view at the specified location.

timestamp. A date and a time, and possibly a number of microseconds (a six- or seven-part value).

TP. Transaction Program

TPN. Transaction program name

transaction program name. The name by which each program participating in an LU 6.2 conversation is known. Normally, the initiator of a connection identifies the name of the program it wants to connect to at the other LU. When used in conjunction with an LU name, it identifies a specific transaction program in the network.

transient data queue. In CICS, a storage area, whose name is defined in the Destination Control Table (DCT), where objects are stored for subsequent internal or external processing.

TSO. Time Sharing Option.

two-phase commit. A protocol used in distributed unit of work to ensure that participating relational database management systems commit or roll back a unit of work consistently.

unit of work. (1) A recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process may involve many units of work as a result of commit or rollback operations. (2) In DRDA, a sequence of SQL commands that the database manager treats as a single entity. The database manager ensures the consistency of data by verifying that either all the data changes made during a unit of work are performed or none of them are performed.

unnamed column. An empty column added to an example table. Like a target table, it is used to combine columns, combine rows, or include constant values in a report.

USA (United States of America) format. A format that represents date and time values as follows:
- Date: mm/dd/yyyy
- Time: hh:mm xM

variation. A data formatting definition specified on a FORM.DETAIL panel that conditionally can be used to format a report or part of a report.

Virtual Storage Extended. An operating system that is an extension of Disk Operating System/ Virtual Storage. A VSE consists of (1) VSE/Advanced Functions support and (2) any IBM-supplied and user-written programs that are required to meet the data processing needs of a user. VSE and the hardware it controls form a complete computing system.

VM. Virtual Machine (IBM operating system). The generic term for the VM/ESA environment.

VSE. Virtual Storage Extended (IBM operating system). The generic term for the VSE/ESA environment.

WAIT. The keyword for an endless-wait-state problem.

Workstation Database Server. The IBM family of DRDA database products on the UNIX and Intel platforms (such as DB2 Universal Database (UDB), DB2 Common Server, DB2 Parallel Edition, and DataJoiner.)

Glossary

Appendix H. Bibliography

The following lists do not include all the books for a particular library. To get copies of any of these books, or to get more information about a particular library, contact your IBM representative.

CICS publications

CICS Transaction Server for OS390

CICS User's Handbook
CICS Application Programming Reference
CICS Application Programming Guide
CICS DB2 Guide
CICS Resource Definition Guide
CICS Problem Determination Guide
CICS System Definition Guide
CICS Intercommunication Guide
CICS Performance Guide

CICS Transaction Server for VSE/ESA

User's Handbook
Application Programming Reference
Application Programming Guide
Resource Definition Guide
Problem Determination Guide
System Definition Guide
Intercommunication Guide
Performance Guide

COBOL publications

COBOL for VSE/ESA Language Reference
COBOL for VSE/ESA Programming Guide

DB2 Universal Database for z/OS publications

DB2 Universal Database for z/OS

Installation Guide
Administration Guide
SQL Reference
Command Reference
Application Programming and SQL Guide
Messages and Codes

Utility Guide and Reference
Reference for Remote DRDA Requesters and Servers

IBM DB2 Server for VSE & VM
Diagnosis Guide and Reference
DB2 Server for VSE Messages and Codes
DB2 Server for VM Messages and Codes
DB2 Server for VSE System Administration
DB2 Server for VM System Administration
DB2 Server for VSE & VM Operation
DB2 Server for VSE & VM SQL Reference
DB2 Server for VSE & VM Application Programming
DB2 Server for VSE & VM Interactive SQL Guide and Reference
DB2 Server for VSE & VM Database Services Utility
DB2 Server for VSE & VM Performance Tuning Handbook

DB2 Universal Database for iSeries
SQL Reference
SQL Programming with Host Languages

DB2 Universal Database
Command Reference
SQL Reference
Message Reference

DB2 DataJoiner
DataJoiner Application Programming and SQL Reference Supplement

DB2 Intelligent Miner
Using DB2 Intelligent Miner for Data

Document Composition Facility (DCF) publications

DCF and DLF General Information

Distributed Relational Database Architecture (DRDA) publications

Every Manager's Guide
Connectivity Guide

Graphical Data Display Manager (GDDM) publications

GDDM General Information
GDDM Base Application Programming Reference
GDDM User's Guide
GDDM/VSE Program Directory
GDDM Messages

GDDM System Customization and Administration

HIgh Level Assembler (HLASM) publications

High-Level Assembler for MVS, VM and VSE Programming Guide
High-Level Assembler for MVS, VM and VSE Language Reference

Interactive System Productivity Facility (ISPF) publications

OS/390

ISPF Planning and Customizing
ISPF Dialog Developer's Guide and Reference

VM

ISPF for VM Dialog Management Guide and Reference

OS/390 publications

JCL

OS/390 MVS JCL Reference
OS/390 MVS JCL User's Guide

Pageable Link Pack Area (PLPA)

OS/390 Extended Architecture Initialization and Tuning
OS/390 SPL: Initialization and Tuning

VSAM

OS/390 VSAM Administration Guide
OS/390 VSAM Catalog Administration Access Method Services

TSO/E

TSO/E Primer
TSO/E User's Guide

SMP/E

OS/390 System Modification Program Extended Messages and Codes
OS/390 System Modification Program Extended Reference
OS/390 System Modification Program Extended User's Guide

OS PL/I publications

OS PL/I Programming Language Reference
OS PL/I Programming Guide

REXX publications

OS/390 environment

TSO/E REXX/MVS User's Guide
TSO/E REXX/MVS Reference

VM environment

System Product Interpreter Reference
REXX/VM User's Guide

VM/ESA publications

VM/ESA Planning and Administration
VM/ESA Command Reference

VSE/ESA publications

Planning
System Utilities
Guide for Solving Problems

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Program Number: 5625-DB2

Printed in USA

GC18-7444-00