Query Management Facility™

Installing and Managing QMF

Version 7 Release 2

GC27-0720-02

Query Management Facility™

Installing and Managing QMF

Version 7 Release 2

GC27-0720-02

Note

Before using this information and the product it supports, be sure to read the general information in Appendix F, "Notices", on page 761.

Fourth Edition (December 2002)

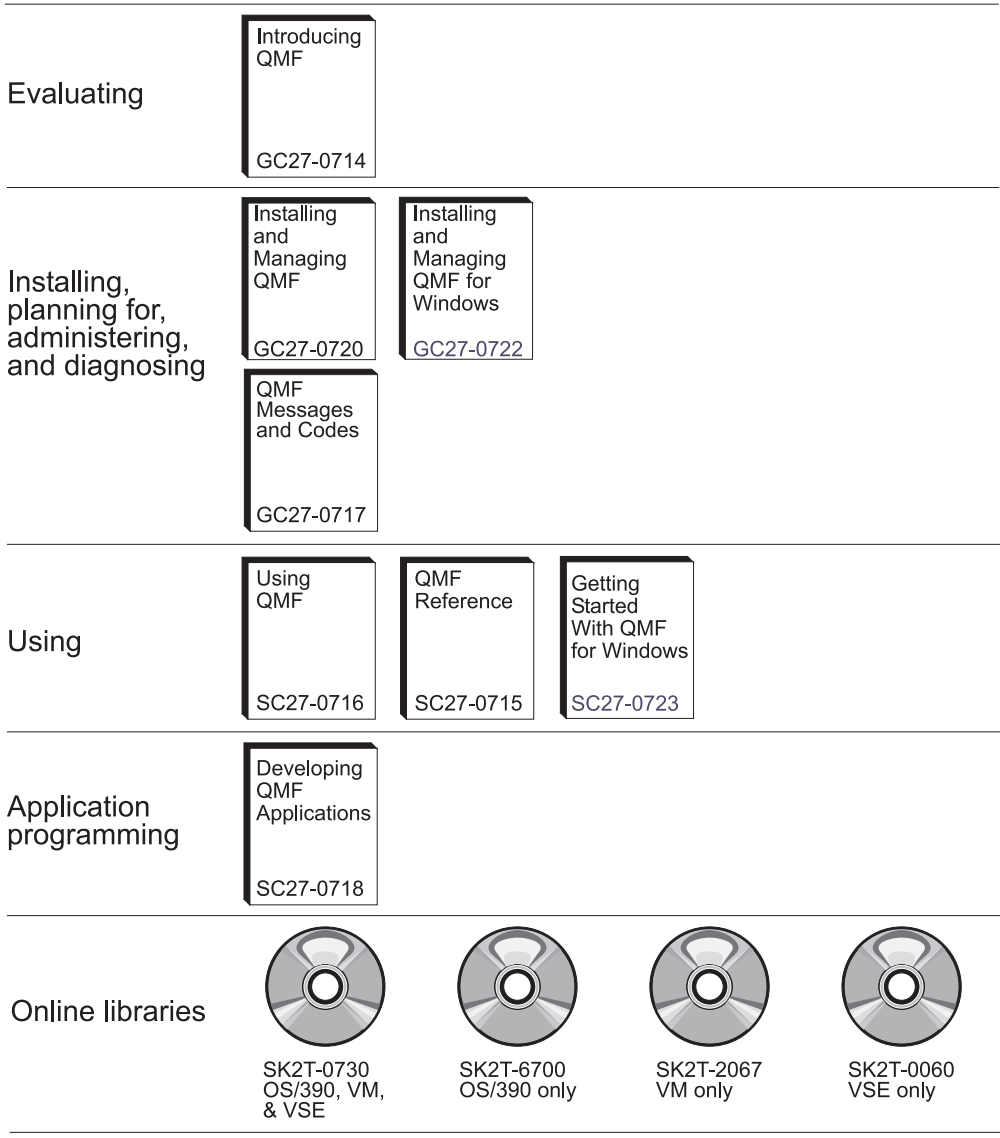
This edition applies to Query Management Facility, a feature of Version 7 Release 1 of DB2 Universal Database Server for OS/390 (DB2 UDB for OS/390), Version 7 Release 1, 5675-DB2, a feature of DB2 Server for VSE and VM Version 7 Release 2, 5697-F42, and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces GC27-0720-01.

Contents

The QMF library

You can order manuals either through an IBM representative or by calling 1-800-879-2755 in the United States or any of its territories.



About this book

This book is intended to help database administrators and systems programmers install and manage the Query Management Facility (QMF) product under Operating System for the Z Architecture (z/OS™) and Operating System/390 (OS/390)®, Virtual Machine/Enterprise Systems Architecture (VM/ESA)®, and Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA)™.

The three Installing and Managing QMF books have been combined into one for Version 7.2. The Installing sections have been kept separate by operating system (z/OS and OS/390, VM, and VSE), while the Managing chapters have been merged. The three Installing sections and the Managing portion each have their own table of contents.

Who should read this book

This book is written for system programmers responsible for installing and managing QMF for use with the IBM DB2 Universal Database for z/OS and OS/390, DB2 for VM, and DB2 for VSE relational databases. 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 Common Server Version 2
- DB2 Parallel Edition for AIX® Version 1.2
- DataJoiner® Version 1.2.1 and Version 2
- DB2 Universal Database Version 5 and higher

What you should know before you begin

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

z/OS or OS/390

On z/OS or OS/390, these components can include:

- Operating systems z/OS or OS/390
- Multiple Virtual Storage/Enterprise System Architecture (MVS/ESA)™ operating system.
- Time Sharing Option (TSO), an environment that supports QMF and its related products.
- Interactive System Productivity Facility (ISPF), a dialog manager for QMF.

- Customer Information Control System (CICS)®, a general-purpose data communication and online transaction processing system. CICS/MVS® provides the interface between QMF and MVS/ESA.
- Graphical Data Display Manager (GDDM)®, which makes it possible for QMF to display panels on the user's screen and create charts.
- DB2, the database manager for QMF.

  DB2 also provides a number of utilities that can run in batch mode or through DB2I (the DB2 interactive facility) on OS/390.
- Data Extract (DXT)™, a facility that can supply the DB2 load utility with data.
- SMP/E (System Modification Program Extended)
- High-level assembly language (HLASM), 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, used to create your own edit codes in PL/I for QMF forms.
- VS COBOL II and COBOL, used to create your own edit codes in COBOL for QMF forms.
- REXX used to create execs that install QMF.

VM

On VM, these components can include:

- The VM/ESA operating system
- Conversational Monitoring System (CMS), an environment that supports QMF and its related products.
- Interactive System Productivity Facility (ISPF), a dialog manager for QMF.
- Graphical Data Display Manager (GDDM), which makes it possible for QMF to display panels on the user's screen and create charts.
- DB2, the database manager for QMF.

  DB2 also provides a number of utilities that can run in batch mode or through ISQL.
- Data Extract (DXT), a facility that can supply the DB2 load utility with data.
- VMSES/E (System Modification Program Extended)
- High-level assembly language (HLASM), 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, used to create your own edit codes in PL/I for QMF forms.
- VS COBOL II and COBOL, used to create your own edit codes in COBOL for QMF forms.
- REXX used to create execs that install QMF.

VSE

On VSE, these components can include:

- Operating system VSE/ESA
- Customer Information Control System (CICS), a general-purpose data communication and online transaction processing system. CICS/VSE provides the interface between QMF and VSE/ESA.
- Graphical Data Display Manager (GDDM), which makes it possible for QMF to display panels on the user's screen and create charts.
- DB2, the database manager for QMF.
- MSHP (System Modification Program)
- High-level assembly language (HLASM), 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, used to create your own edit codes in PL/I for QMF forms.
- VS COBOL II and COBOL, used to create your own edit codes in COBOL for QMF forms.

Publications that discuss these products are listed in Appendix H, "Bibliography", on page 775.

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 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 for OS/390, VM/ESA or VSE/ESA is DB2.

Prerequisite and related information

In addition to this guide, keep the following documents ready during the installation:

- *QMF Program Directory*
- *QMF Preventive Service Planning (PSP)* bucket

The *QMF PSP* bucket documents late-breaking information about the installation.

For a list of QMF publications, see "The QMF library" on page ix. Publications from other IBM product families are found in Appendix H, "Bibliography", on page 775.

How National Language Feature information is represented

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 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 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

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

NLF	NLID	Name that QMF uses for this NLF
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 send your comments

Your feedback helps IBM to provide the most accurate and high-quality information.

Send your comments from the Internet

Visit the QMF Web site at:

`http://www.ibm.com/qmf`

The Web site has a feedback page that you can use to enter and send comments.

Send your comments by e-mail

To comments@vnet.ibm.com. Be sure to include the name of the product, the version number of the product, the name and part number of the book (if applicable). If you are commenting on specific text, include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

Complete the readers' comment form

At the back of the book and return it by mail, by fax (800-426-7773 for the United States and Canada), or by giving it to an IBM representative.

How to order QMF books

You can order QMF documentation either through an IBM representative or by calling 1-800-879-2755 in the United States or any of its territories.

For a list of QMF books, see "The QMF library" on page ix.

Where to next?

To quickly turn to the section for your system's installation, go to the following pages:

- Installing QMF on z/OS and OS/390 Part 1, "Installing QMF on z/OS and OS/390", on page 1
- Installing QMF on VM/ESA Part 2, "Installing QMF on VM/ESA", on page 143
- Installing QMF on VSE/ESA Part 3, "Installing QMF on VSE/ESA", on page 193
- Managing QMF on z/OS and OS/390, VM, and VSE Part 4, "Managing QMF", on page 231

Part 1. Installing QMF on z/OS and OS/390

Chapter 1. Introducing QMF and the Install Process

This chapter introduces the QMF host product. It also provides an overview of how QMF connects to the DB2 Universal Database for OS/390, DB2 Universal Database for z/OS, DB2 Universal Database, Database2 for VM or VSE, Database2 for iSeries databases, and of how QMF is installed.

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 MVS/Enterprise System Architecture (MVS/ESA), and primarily accesses data through DB2 UDB for OS/390. 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.

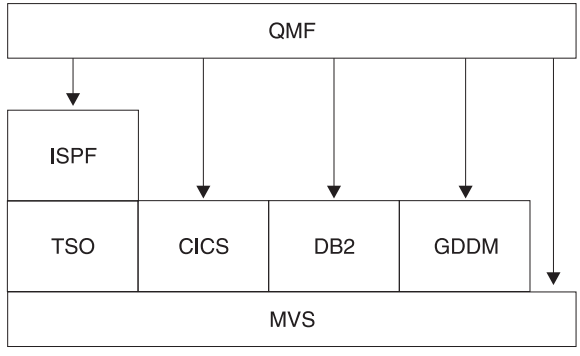


Figure 1. QMF in a Host-Only configuration.

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 OS/390, DB2 for VSE or VM, DB2 for iSeries, or DB2 Universal Database 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 the *DB2 UDB for OS390 SQL Reference*

QMF supports two methods of data access:
- Distributed Relational Database Architecture (DRDA) remote unit of work
- DB2 UDB for OS/390-to-DB2 UDB for OS/390 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 OS/390-to-DB2 UDB for OS/390 distributed unit of work allows you to access other DB2 UDB for OS/390 subsystems using a communications method specific to DB2 UDB for OS/390. DB2 UDB for OS/390 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 OS/390 Distributed Data Facility (DDF) adopted DRDA's data structure beginning with DB2 UDB for OS/390 Version 2.3, DB2 for VSE or VM adopted DRDA's structure in Version 7.1. With remote unit of work, DB2 UDB for OS/390 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.

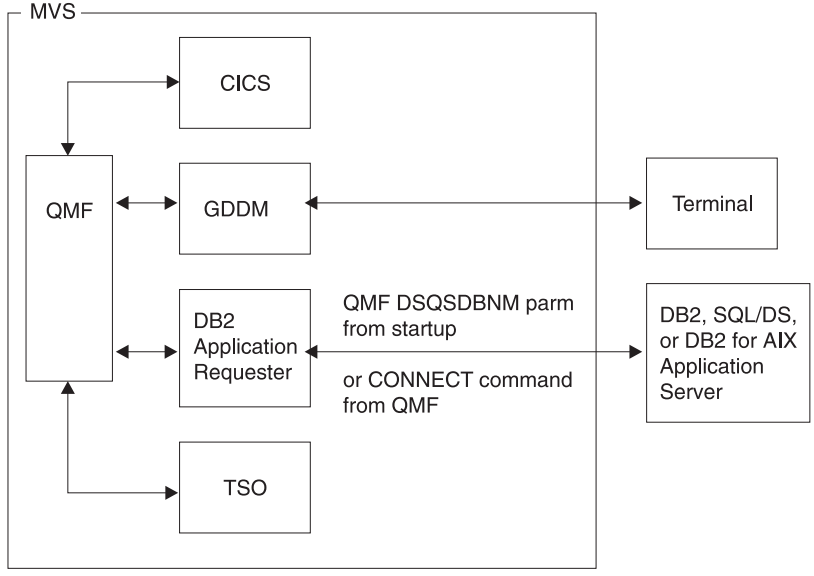


Figure 2. QMF using remote unit of work

DB2 UDB for OS/390-to-DB2 UDB for OS/390 distributed unit of work

This is an early version of distributed unit of work, first introduced in DB2 UDB for OS/390 Version 2.2. It allows access to other DB2 UDB for OS/390 subsystems using a communications method that is private to DB2 UDB for OS/390. With this method you can connect to one location and run one query per unit of work. DB2 UDB for OS/390-to-DB2 UDB for OS/390 distributed unit of work uses an alias or a three-part name to determine the location of the subsystem and to connect to it. QMF, however, requires a minimum level of DB2 UDB for OS/390 Version 2.3 to support this type of data access.

Figure 3 on page 8 shows a DB2 UDB for OS/390-to-DB2 UDB for OS/390 access connection.

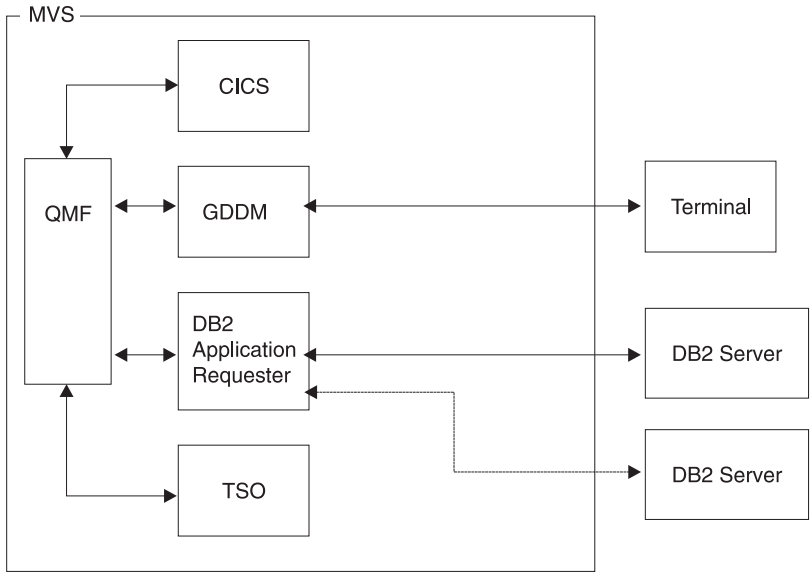


Figure 3. DB2 UDB for OS/390-to-DB2 UDB for OS/390 connection

Overview of the database installation process

Installing QMF on OS/390 involves these object groups:

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

DB2 UDB for OS/390 requirements for QMF

QMF is a DB2 UDB for OS/390 application program that uses standard interfaces to the database. QMF must be installed into at least one DB2 UDB for OS/390 subsystem. Depending on the design of your data network, you may need to install QMF into additional DB2 UDB for OS/390 subsystems.

Prerequisite DB2 UDB for OS/390 knowledge

Because QMF is a DB2 UDB for OS/390 application, you need to understand many of the same concepts as you would to perform a DB2 UDB for OS/390 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 OS390 SQL Reference*

- The terms *application plan*, *DBRM*, *package*, and *bind*

These terms are described in the *DB2 UDB for OS390 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 OS390 Administration Guide* , Volume 2.

- The DB2 UDB for OS/390 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 OS390 Administration Guide* , Volume 2.

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

  For information on subsystems IDs, see the *DB2 UDB for OS390 Administration Guide* , Volume 2.

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 OS/390
- Location name
- Remote unit of work

For definitions and MORE information about these terms, see the *DB2 UDB for OS390 SQL Reference*

DB2 UDB for OS/390 objects created by QMF install

A DB2 UDB for OS/390 system that is accessed by QMF contains a number of object types that are created for QMF during installation.

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 OS/390-to-DB2 UDB for OS/390 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 729.

Database authorization ID Q

Although the DB2 UDB for OS/390 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.

Road maps for the QMF installation process

This section lists the QMF install options types and the DB2 UDB for OS/390 objects created under each install option. For more information about these objects, see Appendix B, "QMF Objects Residing in DB2", on page 729.

- Initial installation or migration
  - Preliminary: Read the program directory and complete the worksheets.
  - Complete the SMPE installation as described in the program directory.
  - Begin a full database install.
- Full database installation
  - Do STEPS 1 and 2.
  - To install in BATCH mode, proceed to STEPS 3 through 16, or
  - To install in FOREGROUND mode, do STEP 3.

  This type of installation creates the following at the local DB2 UDB for OS/390, which is the subsystem where the QMF application plan is bound:
  - QMF target and distribution libraries
  - Two installation packages
  - One QMF installation plan
  - QMF control tables
  - QMF catalog views
  - Table space for QMF SAVE DATA and IMPORT TABLE commands
  - QMF sample tables
  - QMF application packages
  - One QMF application plan

  You need to perform a full install when:
  - It is the initial installation of QMF.
  - It is the only installation of QMF.
  - You need to access more than one local DB2 UDB for OS/390 subsystem from QMF. Perform this type of installation for each additional local DB2 UDB for OS/390 installation.

- Server database installation
  - Do STEPS 1 and 2.
  - To install in BATCH mode, proceed to steps 8 through 16, or
  - To install in FOREGROUND mode, do step 3.

  This type of installation creates:
  - Two installation packages at the DB2 UDB for OS/390 application server
  - One QMF installation plan (using the application server name as the CURRENTSERVER location) at the local DB2 UDB for OS/390 subsystem
  - QMF control tables
  - QMF catalog views
  - Table space for QMF SAVE DATA and IMPORT TABLE commands
  - QMF sample tables
  - QMF application packages

  You need to perform a server database install when you plan to access data that is defined in a different DB2 UDB for OS/390 database. You can run this type of installation on any DB2 UDB for OS/390 subsystem that is accessible from your local DB2 UDB for OS/390 subsystem. Perform the installation from the local DB2 UDB for OS/390 subsystem.

  For information about installing QMF for a Workstation Database Server, see Chapter 7, “Tailoring QMF for Workstation Database Servers”, on page 75.
- Requester database installation
  - Do steps 1 and 2.
  - To install in BATCH mode, run steps 8, 15, and 16, or
  - To install in FOREGROUND mode, run step 3.

  This type of installation creates:
  - Two installation packages at the local DB2 UDB for OS/390 subsystem
  - One QMF installation plan
  - One QMF package (DSQIRDBR)
  - One QMF application plan
  - QMF run-time libraries

  You need to perform a requester database install when you need to access other databases using remote unit of work and you are planning to use this DB2 UDB for OS/390 subsystem as the local DB2 UDB for OS/390 subsystem when running QMF. You can establish a QMF application requester on a DB2 UDB for OS/390 subsystem that is defined on the same OS/390 system where the QMF run-time libraries are installed.

Setting up QMF for remote unit of work

The simplest way to set up DB2 UDB for OS/390 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 OS/390 subsystem on the same OS/390 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 OS/390 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 OS/390 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 UDB for OS/390-to-DB2 UDB for OS/390 distributed unit of work

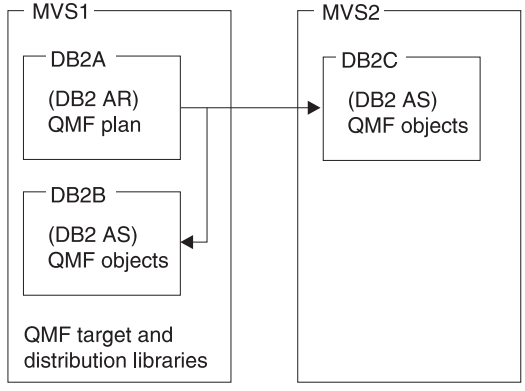
DB2 UDB for OS/390-to-DB2 UDB for OS/390 distributed unit of work access to remote data is mostly transparent to QMF. Therefore, the install process you choose depends on whether you also plan to use remote unit of work. When using both remote unit of work and DB2 UDB for OS/390-to-DB2 UDB for OS/390 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 OS/390 operating system MVS1 has two DB2 UDB for OS/390 Version 2.3 subsystems: DB2A and DB2B. This system is a TSO system; DB2A is an application requester, and DB2B is an application server.
- The OS/390 operating system MVS2 has one DB2 UDB for OS/390 Version 2.3 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 OS/390 operating systems.



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

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 OS/390 and DB2B as the application server. You do not need to log on to MVS2, since the remote installation is run at MVS1.

Chapter 2. Planning for QMF

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 OS/390 and DB2 UDB for OS/390 and all terminals supported by GDDM.

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

Prerequisite software

The following table lists the program products with the minimum release levels required to support QMF for MVS Version 7.2. Later releases that are not available at the QMF Version 7.2 announcement time are not supported unless specifically stated otherwise.

Table 2. Prerequisite software for QMF for OS/390 Version 7.2

Prerequisite software for QMF for OS/390		
Required software	**Version and release**	**Number**
MVS/ESA SP JES2 or	Version 4 Release 2	5695–047
MVS/ESA SP JES3	Version 4 Release 2	5695–048
MVS/Data Facility Product (DFP)	Version 3 Release 1	5665-XA3
Database 2(DB2)	Version 3 Release 1	5685–DB2
GDDM-MVS	Version 2 Release 3	5665-356
For installation only:		
Interactive System Product Facility (ISPF)-MVS	Version 3 Release 5	5685–504
System Modification Program Extended (SMP/E)	Version 1 Release 8	5668-949
For the TSO environment:		
TSO/Extensions (TSO/E)	Version 2 Release 4	5685-025
For the CICS environment:		

Table 2. Prerequisite software for QMF for OS/390 Version 7.2 (continued)

Prerequisite software for QMF for OS/390		
Required software	**Version and release**	**Number**
CICS/ESA or	Version 4 Release 1.1	5655-018
CICS/ESA or	Version 3 Release 1	5683-083
CICS/MVS	Version 2 Release 1.1	5665–403

Table 2. Prerequisite software for QMF for OS/390 Version 7.2

Prerequisite software for QMF for OS/390		
Required software	**Version and release**	**Number**
MVS/ ESA SP JES2 or	Version 5 Release 2	5645–001
MVS/ESA SP JES3	Version 5 Release 2.1	5645-001
DFSMSdfp	Version 1 Release 3	5645-001
Database2 (DB2)	Version 3 Release 1	5685-DB2
GDDM/MVS	Version 3 Release 1.1	5645-001
For installation only:		
Interactive System Product Facility (ISPF)-MVS	Version 4 Release 2.0	5645-001
System Modification Program Extended (SMP/E)	Version 1 Release 8.1	5645-001
For the TSO environment:		
TSO/Extensions (TSO/E)	Version 2 Release 5	5645-001
For the CICS environment:		
CICS/ESA or	Version 3 Release 3	5683–083
CICS/ESA	Version 4 Release 1.1	5655–018

Table 2. Prerequisite software for QMF for OS/390 Version 7.2

Prerequisite software for QMF for MVS/XA		
Required software	**Version and release**	**Number**
MVS/SP-JES2 or	Version 2 Release 2	5740–XC6
MVS/SP-JES3	Version 2 Release 2.1	5665-291
MVS/XA Data Facility Product (DFP)	Version 2 Release 1	5665-XA2
GDDM/MVS	Version 2 Release 1	5665-403
For installation only:		
Interactive System Product Facility (ISPF)-MVS	Version 2 Release 3	5665-319

Table 2. Prerequisite software for QMF for OS/390 Version 7.2 (continued)

Prerequisite software for QMF for MVS/XA		
Required software	**Version and release**	**Number**
System Modification Program Extended (SMP/E)	Version 1 Release 8	5668-949
For the TSO environment:		
TSO/Extensions (TSO/E)	Version 2 Release 1	5685-025
For the CICS environment:		
CICS/MVS	Version 2 Release 1	5665–403

The following table lists the program products with the minimum release levels required to support optional functions for QMF for OS/390 Version 7.2. Later releases that are not available at the QMF Version 7.2 announcement time are not supported unless specifically stated otherwise.

Table 3. Prerequisite software for optional functions for QMF for OS/390 Version 7.2

Product	**Version and release**	**Number**
ISPF-related functions — QMF Document Interface, default editor for QMF EDIT command, display printed report application (DPRE), ISPF command, and DXT/End User Dialogs bridge support:		
ISPF for MVS	Version 3 Release 5	5685-054
Logic in procedures, report calculations and conditions, form column definition, and use of the IBM-supplied command synonyms (DPRE, ISPF, BATCH, LAYOUT):		
TSO/Extensions (TSO/E)	Version 2 Release 4	5685-025
Charts (Interactive Chart Utility:		
GDDM Presentation Graphics Facility (PGF)	Version 2 Release 1.1	5668–812
QMF Document Interface. The following editor is required:		
Personal Services/TSO (PS/TSO)	Release 1	5665–346
Data Extract (DXT). End User Dialogs and the QMF EXTRACT COMMAND:		
Data Extract (DXT)	Version 2 Release 5	5668–788
QMF High Performance Option (HPO):		
ISPF for MVS	Version 3	5685–054
QMF for Windows:		
Microsoft Windows 95 or 98		
Microsoft Windows ME		
Microsoft Windows 2000		

Table 3. Prerequisite software for optional functions for QMF for OS/390 Version 7.2 (continued)

Product	Version and release	Number
Microsoft Windows XP		
Microsoft Windows NT	Version 4.0	
IBM APPC Networking Services for Windows, or	Version 1	
Microsoft SNA Server, or	Version 2	
Novell Netware for SAA, or	Version 2	
Attachmate EXTRA! APPC Client	Version 3 Release 11	
Callable Interface Programs written in the callable interface can be written in:		
IBM C/370 Compiler and	Version 2	5688–187
C/370 Library	Version 2	5688–188
IBM HLASM	Version 1 Release 1 or Release 2	5696–234
VS COBOL II Compiler and Library	Version 1 Release 4	5688–023
VS COBOL II Compiler, Library and Debugging Facility	Version 1 Release 4	5668–958
AD/Cycle COBOL/370	Version 1 Release 1	5688–197
IBM COBOL for MVS and VM	Version 1 Release 2	5688–197
AD/Cycle C/370 Compiler	Version 1 Release 1	5688–216
VS FORTRAN (REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)	Version 2 Release 5	5668–806
OS PL/I	Version 2 Release 3	5668–909
IBM PL/I for MVS and VM	Version 1 Release 1.1	5688-265
REXX: TSO Extensions (TSO/E) (REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)	Version 2 Release 1	5685–025
REXX(REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)	In VM/ESA	
Assembler H	Version 2 Release 1	5668–962

Table 3. Prerequisite software for optional functions for QMF for OS/390 Version 7.2 (continued)

Product	Version and release	Number
IBM C/C++ for MVS/ESA (In conjunction with Language Environment for MVS and VM (MVS feature)).	Version 3	5655–121
User Edit Routines can be written in:		
IBM HLASM	Version 1	5696–234
VS COBOL II Compiler and Library	Version 1 Release 4	5688–023
COBOL/370 Compiler and Library	Version 1 Release 1	5688–197
IBM COBOL for MVS and VM	Version 1 Release 2	5688–197
VS COBOL II Compiler and Library	Version 1 Release 3.1	5688–023
VS COBOL II Compiler, Library and Debugging Facility	Version 1 Release 3.1	5668–958
OS PL/I	Version 2 Release 3	5668–909
IBM PL/I for MVS and VM	Version 1 Release 1.1	5688-265
Assembler H or standard assembler	Version 2 Release 1	5668–962
Governor Exit Routine		
IBM HLASM	Version 1	5696–234
Assembler H or standard assembler	Version 2 Release 1	5668–962
Remote Unit of Work (OS/390)		
Connection to remote DB2 on OS/390 DRDA Application Server:		
At the local DB2 for OS/390 location:		
DB2 for MVS	Version 3 Release 1	5685–DB2
QMF for OS/390	Version 7 Release 2	5675-DB2
At the remote DB2 database:		
DB2 for MVS	Version 3 Release 1	5685–DB2
QMF for OS/390	Version 7 Release 2	5675-DB2
Connection to remote DB2 on VM DRDA Application Server:		
At the local DB2 for MVS/ESA location:		

Table 3. Prerequisite software for optional functions for QMF for OS/390 Version 7.2 (continued)

Product	Version and release	Number
DB2 for MVS	Version 3 Release 1	5685–DB2
QMF for OS/390	Version 7 Release 2	5675-DB2
At the remote DB2 for VM/ESA or VSE/ESA database:		
SQL/DS for VM	Version 3 Release 5	5688–103
SQL/DS for VM	Version 3 Release 3	5706–255
Connection to remote DB2 on VSE DRDA Application Server:		
At the local DB2 for OS/390 location:		
DB2 for MVS	Version 3 Release 1	5685–DB2
QMF for OS/390	Version 7 Release 2	5675-DB2
At the remote DB2 for VM or VSE database:		
SQL/DS	Version 3 Release 5	5688–103
DB2 for VSE	Version 6	5648–061
Connection to DB2 PE, DataJoiner, Common Server:		
At the local DB2 for OS/390 location:		
DB2 for MVS	Version 3 Release 1 with PTF UP75959 and PTF UN54601	5685–DB2
QMF for OS/390	Version 7 Release 2	5675-DB2
At the remote database configured for APPC communications:		
DB2 Parallel Edition for AIX or	Version 1 Release 2	5765–328
DataJoiner for AIX or	Version 1 Release 2	84H1212
DB2 for Windows NT	Version 2 Release 1	53H7474
DB2 for AIX or	Version 2 Release 1	41H2128
DB2 for HP-UX or	Version 2 Release 1	10H2366
DB2 for Solaris or	Version 2 Release 1	
DB2 for SCO OpenServer or	Version 2 Release 1	79H5359
DB2 for SINIX	Version 2 Release 1	79H4133

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 2.8 MB.
- QMF modules that must run in a 24-bit addressing mode require 52 KB.
- Minimum storage for users to execute QMF queries and hold QMF report data is between 0.5 and 1.0 MB. Your specific requirements can be larger depending on the size of your report and the report formatting options used.

As an example, if you run in a standard TSO environment with ISPF and GDDM, you need approximately 6.0 MB of storage.

You might require more than 1 MB of storage if you use competing options for a report, or if a query returns a large amount of data. You can allocate storage for both purposes above 16 MB.

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

OS/390 storage

You need 0.5 to 1 MB of storage to run QMF. Additional storage is required for other applications. For example, if you run QMF in a standard TSO environment with ISPF and GDDM, you need approximately 6 MB of storage.

Most of the QMF modules are reentrant and can be loaded into EPLPA. One 52 KB 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. You might use 16 MB plus 1 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

After installation, the library QMF720.SDSQLOAD contains the load modules for the QMF program.Table 4 shows the modules you can move into link pack area libraries to enhance performance.

Table 4. 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 5 describes the modules that cannot be placed in the PLPA or EPLPA.

Table 5. TSO Modules that can't reside in the PLPA or EPLPA

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

Table 5. 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 SMP/E 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.

Estimated DASD space (in cylinders) for the SMP/E data sets is shown in Table 6.

Table 6. DASD space for SMP/E data sets

DDname	3380	3390	9345
SMPSCDS	1	1	1
SMPCSI	8	8	8
SMPLOG	1	1	1
SMPMTS	1	1	1
SMPPTS	1	1	1
SMPSTS	1	1	1

Estimating space for distribution libraries

The QMF distribution libraries and their estimated DASD space (in tracks) are shown in Table 7 on page 24.

Table 7. DASD space for QMF distribution libraries

DSNAME	Content	3380	3390	9345
QMF720ADSQOBJ	QMF object modules	11	11	9
QMF720.ADSQMACE	QMF install procedures	15	15	13
QMF720.ADSQDBMD	Database request modules	1	1	1
QMF720.ADSQPMSE	QMF ISPF panels	1	1	1

Estimating target library size

Table 8 shows estimates for your required DASD space (in cylinders) for the target libraries.

Table 8. DASD space for QMF target libraries

DSNAME	Content	3380	3390	9345
QMF720.SDSQLOAD	QMF load modules	8	8	7
QMF720.SDSQEXIT	QMF user exits	1	1	1
QMF720.SDSQSAPE	IVP, sample queries	17	17	15
QMF720.SDSQDBRM	QMF DBRMs	1	1	1
QMF720.SDSQPLBE	ISPF panels for QMF	1	1	1
QMF720.SDSQCLTE	Sample QMF CLIST	2	2	2
QMF720.SDSQSLBE	Sample ISPF skeletons	1	1	1
QMF720.SDSQMLBE	Sample ISPF messages	1	1	1
QMF720.SDSQEXCE	TSO/E REXX procs	1	1	1
QMF720.SDSQUSRE	Sample user exit routines	1	1	1

After you allocate space for your libraries, you can use SMP/E to install QMF.

Estimating space for user data sets

Estimated DASD space required (in cylinders) for the QMF user libraries is shown in Table 9.

Table 9. DASD space for QMF user data sets

DSNAME	Content	3380	3390	9345
QMF720.DSQMAPE	GDDM map group files	1	1	1
QMF720.DSQCHART	GDDM samples chart files	1	1	1
QMF720.DSQUCFRM	GDDM/CICS samples chart forms files (expanded in VSAM format)	1	1	1

Table 9. DASD space for QMF user data sets (continued)

DSNAME	Content	3380	3390	9345
QMF720.DSQPVARE	QMF message help panels (expanded in sequential format)	N/A	6	6
QMF720.DSQPNLE	QMF message help panels (expanded in VSAM format)	N/A	10	9
QMF720.GDDM.ADMF	GDDM/CICS data set (in VSAM format)	1	1	1

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 preventive service planning (PSP) bucket, SUBSET: QMFMVS under UPGRADE QMF720. 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.

Planning for QMF under CICS

You need to complete installation, tailoring, and testing of 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/OS390 Intercommunication Guide*.

Planning for QMF

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* and *GDDM/MVS Installation, Testing, and Servicing*guides.

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.

Planning for QMF for DB2 UDB for OS/390 for AIX

Customizing QMF to work with a DB2 UDB for OS/390 for AIX server requires some changes both on the host and the server.

From OS/390, QMF uses the distributed data facility (DDF) of DB2 UDB for OS/390 to access distributed data that resides in a DB2 UDB for OS/390 for AIX database. The DDF of DB2 UDB for OS/390 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 on OS/390 is in *DB2 UDB for OS390 Administration Guide* , Volume 1.

From DB2 UDB for OS/390, the communications database (CDB) tables are used to control access between remote database management systems. If you plan to use DB2 UDB for OS/390 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 OS390 Administration Guide* , Volume 2 and *DB2 UDB for OS390 SQL Reference*.

At the DB2 UDB for OS/390 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 OS/390 for OS/390 DRDA application requester and the DB2 UDB for OS/390 for AIX DRDA application server. For more information about installing your DB2 UDB for OS/390 for AIX server, refer to the installation instructions for that DB2 UDB for OS/390 for AIX server.

For more information about the installation from OS/390, of QMF objects into DB2 UDB for OS/390 for AIX, and about the prerequisites for the installation, see Chapter 7, "Tailoring QMF for Workstation Database Servers", on page 75.

Complete the worksheets

Table 10, displays the parameters you need to provide values for during the QMF installation. Use them as worksheets.

Table 10. QMF Installation Parameters (Version 7 Worksheet part 1

PARAMETER	VALUE
Location name	
Target Library Prefix (Default = QMF720)	
Distribution Library Prefix (Default = QMF720)	
Target Library Volume (Default = xxxxxx)	
Distribution Library Volume (Default = xxxxxx)	
SMP/E Data Set Prefix (Default = IMSVS)	
Local DB2 UDB for OS/390 Subsystem ID (Default=DSN)	
Local DB2 UDB for OS/390 Release Level (Default=V3R1)	
Local DB2 UDB for OS/390 Exit Library (Default=DSN710.SDSNEXIT)	
Local DB2 UDB for OS/390 Load Library (Default=DSN710.SDSNLOAD)	
Communications database installed at local DB2 UDB for OS/390	yes or no
Gather the following information if the communications database is installed at the local DB2 UDB for OS/390 subsystem:	
Scope of installation	F (Full database), S (Server database), or R (Requester database)
Gather the following information, if the scope of the database install is not "S"(Server database):	
Customize QMF runtime libraries	yes or no

Table 10. QMF Installation Parameters (Version 7 Worksheet part 1 (continued)

PARAMETER	VALUE
QMF application plan ID (Default=QMF720)	

PARAMETER	VALUE
Gather the following information if the scope of the database install is "S" (Server database):	
DB2 Universal Database for OS/390 server location in remote DB2 Universal Database for OS/390 subsystem? (Default=NO)	yes or no
Gather the following information, if the scope of the database install is "F"(Full database), or the scope of the database install is "S"(Server database), and the server database is the same as the local subsystem.	
DB2 Universal Database for OS/390 user catalog (ICF) (Default=DSNC7101.USER.CATALOG)	
DB2 Universal Database for OS/390 user catalog password	
QMF tablespace catalog alias (Default=QMFDSN)	
QMF tablespace catalog password (for QMF control tables)	
QMF tables volume	
DB2 Universal Database for OS/390 default punctuation	, (comma) or . (period)
Previous QMF level (migration installs only)	V3R1, V3R1M1, V3R2, V3R3, 6R1, V7R1, or none
Gather the following information when the scope of the database install is "S" (Server database) and the server database is different from the local DB2 Universl Database for OS/390 subsystem.	
DB2 Universal Database for OS/390 server location name	
DB2 Universal Database for OS/390 server in another operating system?	yes or no
DB2 Universal Database for OS/390 user catalog (ICF) for server (Default=DSNC7101.USER.CATALOG)	
DB2 Universal Database for OS/390 user catalog password	
QMF tablespace catalog alias at server (Default=QMFDSN)	
QMF tablespace catalog password (for QMF control tables)	
QMF tables volume at server	
DB2 Universal Database for OS/390 default punctuation at server	, (comma) or . (period)
Previous QMF level at server (migration installs Only)	V3R1, V3R1M1, V3R2, V3R3, V6R1, V7R1, or none

PARAMETER	PRIMARY	SECONDARY
Gather the following information, if the previous QMF level is not NONE, and the scope of the database install is not "R".		
QMF control table tablespace 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)	_____ _____ _____ _____ _____ _____ _____ 100_____ _____	_____ _____ _____ _____ _____ _____ _____ _20_____ _____
Table Index Sizes: (in 1K units) Table Index name Default size (primary, secondary) - Q.OBJECT_DIRECTORYX (see Note) (100,20) - Q.OBJECT_REMARKSX " (100,20) - Q.OBJECT_OBJDATAX " (100,20) - Q.PROFILEX (100,20) - Q.COMMAND_SYNONYMSX " (100,20)	_____ _____ _____ _____ _____	_____ _____ _____ _____ _____
Determine the following, if applicable:		
Install QMF jobs in the foreground or tailor the JCL files yourself and run each job in batch?	foreground or batch	
Do you have fixed or variable-length CLIST libraries?	variable or fixed	
Do you have fixed or variable-length EXEC libraries?	variable or fixed	
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.

Chapter 3. Submitting QMF Batch Install Jobs

This chapter describes how to install QMF in a batch environment.

Step 4—Install QMF panels

DSQ1EPNL uses two data sets (DSQPVARE and DSQPNLE) that were created in Chapter 2, "Planning for QMF", on page 15 when DSQ1EJAL was run.

DSQ1EPNL copies expanded versions of QMF panels to the panel file QMF720.DSQPNLE.

1. Edit QMF720.SDSQSAPE(DSQ1EPNL).
2. Verify or change the following values in the instream procedure of the job:

```
//DSQ1PNL PROC RGN='2048K',  Job-step region size
//     QMFTPRE='QMF720 '      Prefix for QMF  target libraries
//     LKEY='E'                                  Language identifier
```

3. Submit job QMF720.SDSQSAPE(DSQ1EPNL).
4. Check for a return code of 0.

Step 5—Install QMF/GDDM map groups

DSQ1EMAP copies expanded versions of certain GDDM map groups to a map group library named QMF720.DSQMAPE. The map groups are copied from the source library QMF720.SDSQSAPE.

1. Edit DSQ1EMAP.
2. Verify and change if necessary these parameters in the instream procedure of the job:

```
//DSQ1MAP PROC RGN='2048K',   Job-step region size
//     QMFTPRE='QMF720 ',      Prefix for QMF  target libraries
//     MAPID=
```

3. Submit job QMF720.SDSQSAPE(DSQ1EMAP).
4. Check for a return code of 0.

Step 6—Install QMF/GDDM sample chart forms

DSQ1CHRT copies expanded versions of GDDM chart-form files from the library QMF720.SDSQSAPE to the library QMF720.DSQCHART.

1. Edit DSQ1CHRT.
2. Verify or change the default values for the input parameters in the job's instream procedure.

```
//DSQCHRT PROC RGN='2048K',  Job-step region size
//    QMFTPRE='QMF720 ',      Prefix for QMF  target libraries
//    CHART=
```

3. Submit job QMF720.SDSQSAPE(DSQ1CHRT).
4. Check for a return code of 0.

Step 7—Convert REXX exec and CLIST records

This step converts REXX exec and CLIST records from fixed to variable length. Two jobs are used for the conversion. The first, DSQ1EJVE, converts QMF REXX exec records from fixed length to variable length. The second, DSQ1EJVC, converts QMF CLIST records from fixed length to variable length.

Converting REXX exec records

The QMF EXEC library contains fixed-length records. It can be concatenated only to other exec libraries that have fixed-length records. If the library contains variable-length records, you must create a copy of the QMF library with variable-length records.

1. Check the following:
   - If other exec libraries have fixed-length records, skip this step.

   Later in the install, you will allocate the QMF exec library (QMF720.SDSQEXCE.VB) as a SYSEXEC data set by concantenating the library to other exec libraries.

   Example

   In the following JCL (from DSQ1EINV), the library QMF720.SDSQEXCE is concatenated to an exec library named SYS2.EXEC:

   ```
   //SYSEXEC  DD DSN=SYS2.EXEC,DISP=SHR
   //         DD DSN=QMF720.SDSQEXCE,DISP=SHR
   ```

   For more information, see the *Developing QMF Applications* manual.
2. Edit QMF720.SDSQSAPE(DSQ1EJVE).

   Change the serial number of the volume for the copy of the library.

   ```
   //DSQTEVB.SYSUT2 DD DISP=(NEW,CATLG),UNIT=SYSDA,
   //    SPACE=(8800,(400,50,25)),VOL=SER=XXXXXX,
   //    DCB=(RECFM=VB,LRECL=84,BLKSIZE=8800)
   ```
3. Change the job statement to conform to your installation.

   ```
   //DSQ1EJVE JOB (ACCT),NAME,
   //    CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),
   //    USER=Q,PASSWORD=Q
   ```
4. Verify and change, if necessary, the value of the QMFTPRE parameter in the instream procedure of the job.

```
//DSQTEVB PROC RGN='2048K',
//    QMFTPRE='QMF720 ',   Prefix for QMF  libraries
//    CLIST=                                    Leave blank;
//*                                    used when the procedure is called
```

5. Submit job QMF720.SDSQSAPE(DSQ1EJVE).
6. Check for a return code of 0.

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

Converting CLIST records

The QMF CLIST library contains fixed-length records. It can be concatenated to only other CLIST libraries that have fixed-length records. If they contain variable-length records, you must create a copy of the QMF library with variable-length records.

1. Check the following:
   - If other CLIST libraries have fixed-length records, skip this step.
   - If your installation uses variable-length records CLIST libraries, continue with this step. It creates a CLIST library that contains variable-length records named QMF720.SDSQCLTE.VB. Use this new CLIST for your SYSPROC concatenation. (QMF720.SDSQCLTE should remain fixed-block because both the QMF install process and SMP/E require a fixed-block CLIST for processing updates.)

   Later in the install, you must allocate the QMF CLIST library (QMF720.SDSQCLTE) as a SYSPROC data set. To do this, concantenate the library to other CLIST libraries.

Example

In the following JCL (from DSQ1EINV), the library QMF720.SDSQCLTE is concatenated to a CLIST library named SYS2.CLIST:

```
//SYSPROC  DD  DSN=SYS2.CLIST,DISP=SHR
//         DD  DSN=QMF720.SDSQCLTE,DISP=SHR
```

2. Edit QMF720.SDSQSAPE(DSQ1EJVC).
3. Verify or change the job statement to conform to your installation.

```
//DSQ1EJVC JOB (ACCT),NAME,
//      CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),
//      USER=Q,PASSWORD=Q
```

4. Verify or change the value of the QMFTPRE parameter in the job's instream procedure.

```
//DSQTIVB PROC RGN='2048K',
//    QMFTPRE='QMF720 ',   Prefix for QMF  libraries
//    EXEC=                                   Leave blank;
//*                      used when the procedure is called
```

5. Change the serial number of the volume for the copy of the library.

```
//DSQTIVB.SYSUT2 DD DISP=(NEW,CATLG),UNIT=SYSDA,
//     SPACE=(8800,(400,50,25)),VOL=SER=XXXXXX,
//     DCB=(RECFM=VB,LRECL=84,BLKSIZE=8800)
```

6. Submit job QMF720.SDSQSAPE(DSQ1EJVC).
7. Check for a return code of 0.

Preparing QMF as a DB2 universal database for OS/390 application

In this series of steps, you:
- Create DB2 UDB for OS/390 resources.
- Make these resources available to DB2 UDB for OS/390.
- Bind QMF to DB2 UDB for OS/390.

If you have an earlier release of QMF installed in the DB2 UDB for OS/390 subsystem, some of these resources are already available.

If you are installing QMF into a DB2 for OS/390 Version 7 (or higher) database, be sure that the database Application Encoding installation parameter for bind of plans and packages is set to either EBCDIC or an EBCDIC ccsid.

In "Step 8—Binding QMF install programs to DB2 UDB for OS/390" you edit and bind two DB2 UDB for OS/390 application programs. In the remaining steps, you run TSO batch jobs (through the program IKJEFT01) and use the output of "Step 8—Binding QMF install programs to DB2 UDB for OS/390" to run DB2 UDB for OS/390 statements. Most of the components for these steps are members of the library QMF720.SDSQSAPE or QMF720.SDSQLOAD. For these steps that run TSO batch, check the step completion codes in the system messages. Completion messages can be found in the SYSTSPRT or the SYSTERM output, as indicated.

Each substep can be restarted, since all of the changes to the DB2 UDB for OS/390 database remain uncommitted until the end of the job.

Step 8—Binding QMF install programs to DB2 UDB for OS/390

This step binds programs DSQDBSQL and DSQDBINS to DB2 UDB for OS/390. The paralleling application plan from the bind is named DSQIN720.

1. Edit QMF720.SDSQSAPE(DSQ1BSQL).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQ1BSQL PROC RGN='2048K',        Job-step region size
//     QMFTPRE='QMF720 ',            Prefix for QMF  target libraries
//     DB2EXIT='DSN710 .SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710 .SDSNLOAD'  DB2 UDB for OS/390 program
                                   library name
```

The job does its work through a series of DB2 UDB for OS/390 statements. The statements are members of the library QMF720.SDSQSAPE.

3. Submit job QMF720.SDSQSAPE(DSQ1BSQL).
4. Check that you received a return code of 0.

   Do not proceed if the return code is other than zero. Examine SYSTSPRT for error messages. Perform corrective actions and then rerun the job.

   If you are doing a Requester database installation, go to "Step 15—Bind QMF application plan to DB2 UDB for OS/390" on page 44; otherwise, continue to the next step.

Step 9—Create QMF control tables

This step creates eight QMF control tables and six catalog views. For more information about these tables and views, see Appendix B, "QMF Objects Residing in DB2", on page 729.

Converting QMF control table indexes to type 2:

If you are running DB2 for MVS/ESA Version 4 or above, the QMF control table indexes will be migrated or created as TYPE 2 indexes. Run the following query to determine the TYPE of your QMF indexes.

```
SELECT  NAME,CREATOR,TBNAME,TBCREATOR,INDEXTYPE FROM  SYSIBM.SYSINDEXES
     WHERE CREATOR = 'Q'
```

If INDEXTYPE is equal to ' ' (blank), the migration jobs will alter the indexes to change them to TYPE 2. After migration, the indexes are left in recover pending state. The index changes do not take place until the indexes are recovered, loaded, or reorganized. Run the DB2 utility REBUILD INDEX (or RECOVER INDEX if you are using DB2 Version 5 or earlier) to complete conversion of the QMF indexes.

If you want to fall back to DB2 R310, you must to convert the indexes back to TYPE 1 prior to falling back. Use the 'ALTER INDEX' SQL statement. Refer to *DB2 UDB for OS390 SQL Reference* for the syntax.

Tips for remote unit of work

If you want to access tables and views at remote DB2 UDB for OS/390 locations, you must run the install job to create the QMF catalog views at each remote location.

Which job you run depends upon whether you are migrating from an earlier QMF version or not. It also varies according to what QMF version and release is in the DB2 UDB for OS/390 subsystem you have selected for QMF Version 7.2.

QMF level in DB2 UDB for OS/390 subsystem
Follow this procedure

QMF Version 7.2
"Step 10— Create a table space for the QMF IVP" on page 41 Skip this step; none of the control tables must be altered.

QMF Version 3.x, QMF Version 6
"Migrating from QMF Version 7.1, Version 6, and Version 3 Release 3.0, 2.0, 1.1, 1.0"

QMF Version 2.4
"Migrating from QMF Version 2.4" on page 37

QMF is new
"Creating control tables without a previous QMF release" on page 39

Migrating from QMF Version 7.1, Version 6, and Version 3 Release 3.0, 2.0, 1.1, 1.0

Run this step if the DB2 UDB for OS/390 subsystem you selected for QMF Version 7.2 currently contains any of the following QMF versions, releases, and migrates: Version 7.1, Version 6.1, all releases of Version 3, DSQ1TBJ0 migrates for Version 7.1, Version 6.1, and all releases of Version 3 control tables to Version 7.2.

Skip this step if you are not migrating from Version 7.1, Version 6, or Version 3.

1. Edit QMF710.SDSQSAPE(DSQ1TBJ0).
2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1TBJ0 PROC RGN='2048K',        Job-step region size
//     QMFTPRE='QMF720 ',            Prefix for QMF  target libraries
//     DB2EXIT='DSN710 .SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710 .SDSNLOAD'   DB2 UDB for OS/390 program
                                    library name
```

3. Submit job QMF710.SDSQSAPE(DSQ1TBJ0).
4. Check that you received 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 rerun this job.

Migrating from QMF Version 2.4

Run this step if the DB2 UDB for OS/390 subsystem you selected for QMF Version 7.2 currently contains QMF Version 2.4.

This install step contains three jobs:
- DSQ1TBD1 stops the index space
- DSQ1TBA1 allocates VSAM files
- DSQ1TBJ0 recreates Control Tables

Stopping the index space (DSQ1TBD1)

This job stops the QMF profile index space so that it can be redefined in the next step.

1. Determine whether the server is in the local DB2 UDB for OS/390. Run this job only if the server is in the local DB2 UDB for OS/390.

   Issue the STOP command in QMF720.SDSQSAPE(DSQ1TBD1) to stop the index space at the server.
2. Edit QMF720.SDSQSAPE(DSQ1TBD1).
3. Verify the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1TBD1 PROC RGN='2048K',        Job-step region size
//     QMFTPRE='QMF720 ',            Prefix for QMF  target libraries
//     DB2EXIT='DSN710 .SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710 .SDSNLOAD'   DB2 UDB for OS/390 program
                                    library name
```

4. Submit QMF720.SDSQSAPE(DSQ1TBD1).
5. Check that you received a return code of 0. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and then rerun this job.

Allocating the VSAM Files (DSQ1TBA1)

This job allocates VSAM files for the QMF index space for Q.PROFILEX. The job works through a series of IDCAMS statements.

1. Tailor DSQ1TBA1 for distributed data.

   If DB2 UDB for OS/390 is Version 2 Release 3 and the server system is remote (not in local system), you must insert a /*ROUTE XEQ JCL statement for JES2 and //*ROUTE XEQ JCL statement for JES3 after the jobcard. These statements are required. You must allocate the alias and VSAM data sets at the remote server system by executing this job through the ROUTE card at the remote server system.
2. Edit QMF720.SDSQSAPE(DSQ1TBA1).

3. Verify the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1TBA1 PROC RGN='2048K',  Job-step region size
//          QMFTPRE='QMF720 '  Prefix for the target libraries
```

4. Submit QMF720.SDSQSAPE(DSQ1TBA1).
5. Check for a return code of 0.

   Examine SYSPRINT for error messages. You can ignore the following message if you receive it on the DELETE and PURGE of the cluster:

```
IDG3012I Entry QMFDSN.DSNDBC.DSQDBCTL.PROFILEX I001,A001.
```

Recreating control tables

DSQ1TBJ0 alters a control table, drops and recreates a control table index, creates the Q.DSQ_RESERVED table space and control table, and creates the QMF catalog views. The job works through a series of DB2 UDB for OS/390 statements and has several members.

1. Edit QMF720.SDSQSAPE(DSQ1TBJ0).
2. Verify the installation parameters in the instream procedure of the job matches your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1TBD1 PROC RGN='2048K',        Job-step region size
//     QMFTPRE='QMF720 ',            Prefix for QMF  target libraries
//     DB2EXIT='DSN710 .SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710 .SDSNLOAD'   DB2 UDB for OS/390 program
                                    library name
```

3. Determine whether your user catalog is password protected. If it is, add the password clause to the STOGROUP statement in member DSQ1VSTB.

```
CREATE STOGROUP PASSWORD(password)
```

4. Submit QMF720.SDSQSAPE(DSQ1TBJ0).
5. Check that you received 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 this job.

Recover indexes converted to type 2

If your indexes were altered to TYPE 2 as a result of running DSQ1TBJ0, they must be recovered, loaded or reorganized.

If you are uncertain whether indexes need to be rebuilt, follow these steps:

1. Review the output of job DSQ1TBJ0. If the final return code was 4, search the job output for the string `ALTER INDEX`. If you find any occurrences of `ALTER INDEX`, perform the next step.
2. Run the following DB2 command:

```
-DISPLAY DATABASE (DSQDBCTL) SPACENAM (*)
```

If any indexes (TYPE=IX) show a STATUS of RECP or RW,RECP, rebuild the indexes.

Depending on your DB2 for OS/390 release, the following DB2 utility job streams must be run:

Version 5 and lower

```
RECOVER INDEX(ALL) TABLESPACE DSQDBCTL.DSQTSCT1
RECOVER INDEX(ALL) TABLESPACE DSQDBCTL.DSQTSCT2
RECOVER INDEX(ALL) TABLESPACE DSQDBCTL.DSQTSCT3
RECOVER INDEX(ALL) TABLESPACE DSQDBCTL.DSQTSPRO
RECOVER INDEX(ALL) TABLESPACE DSQDBCTL.DSQTSSYN
```

Version 6 and above

```
REBUILD INDEX (Q.OBJECT_DIRECTORYX)
REBUILD INDEX (Q.OBJECT_REMARKSX)
REBUILD INDEX (Q.OBJECT_OBJDATAX)
REBUILD INDEX (Q.PROFILEX)
REBUILD INDEX (Q.COMMAND_SYNONYMSX)
```

Creating control tables without a previous QMF release

Run this step if the DB2 UDB for OS/390 subsystem for QMF does not contain an earlier release of QMF.

This step contains two install jobs:

- DSQ1TBAJ allocates alias. (Skip this if your system already has the alias defined.)
- DSQ1TBLJ creates and loads QMF control tables and catalog views.

Allocating alias and VSAM files

DSQ1TBAJ allocates the alias for the QMF control tables and views. The job does its work through a series of IDCAMS statements.

1. Edit QMF720.SDSQSAPE(DSQ1TBAJ).
2. Verify the installation parameters in the instream procedure of the job and the job steps match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

   ```
   //DSQ1TBAJ PROC RGN='2048K',  Job-step region size
   //         QMFTPRE='QMF720 ',  Prefix for QMF target libraries
   ```

3. Tailor the job for distributed data, if applicable.

   If DB2 UDB for OS/390 is at Version 2 Release 3 and the server system is remote (not in local system), you must insert a /*ROUTE XEQ JCL statement for JES2 and //*ROUTE XEQ JCL statement for JES3 after the jobcard. These statements are required. You must allocate the alias and

VSAM data sets at the remote server system by executing this job through the ROUTE card at that remote server system.

DSQ1TBLR

The job step defines the alias for QMF in the DB2 UDB for OS/390 VSAM catalog. It contains this statement:

```
DEFINE ALIAS   -
 (NAME('QMFDSN') RELATE('DSNC7101.USER.CATALOG'))
```

4. Verify that this statement matches your tailoring specifications.
5. Submit QMF720.SDSQSAPE(DSQ1TBAJ).
6. Check for a return code of 0.

   If the first step fails, examine SYSPRINT for error messages and rerun the job after you correct the error. If the second step fails, restart DSQ1VSTA. If you receive an SQLCODE 203 on the location, then see the section Appendix A, "Miscellaneous", on page 723.

Creating and loading QMF control tables and catalog views

DSQ1TBLJ creates and loads QMF control tables and catalog views by working through a series of DB2 UDB for OS/390 statements.

1. Edit QMF720.SDSQSAPE(DSQ1TBLJ).
2. Verify the installation parameters in the instream procedure of the job and the job steps match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

   ```
   //DSQ1TBLJ PROC RGN='2048K',        Job-step region size
   //     QMFTPRE='QMF720 ',            Prefix for QMF  target libraries
   //     DB2EXIT='DSN710 .SDSNEXIT',  Exit DB2 UDB for OS/390 library name
   //     DB2LOAD='DSN710 .SDSNLOAD'   DB2 UDB for OS/390 program
                                       library name
   ```

3. Determine whether your user catalog is password protected. If it is, add the password clause to the STOGROUP statement in member DSQ1VSTB.

   ```
   CREATE STOGROUP PASSWORD(password)
   ```

4. Verify that the installation parameter for the QMF Catalog Alias (default **QMFDSN**) matches your tailoring specifications in the following members:
   - DSQ1VSTD
   - DSQ1TBLB
   - DSQ1TBLI
   - DSQ1TBLU
   - DSQ1TBLE
   - DSQ1TBLN
   - DSQ1TBLG
5. Submit QMF720.SDSQSAPE(DSQ1TBLJ).

6. Check that you received 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 10— Create a table space for the QMF IVP

DSQ1STGJ creates a table space named DSQDBDEF.DSQTSDEF for the QMF installation verification procedure (IVP). Before it creates the table space, it creates the storage group (DSQSGDEF) and the database (DSQDBDEF) for this table space. After installation, you can use this table space for the tables your users create.

Skip this step if the table space already exists in the DB2 UDB for OS/390 subsystem that you selected for QMF Version 7.2. If you attempt to create a second table space, you will receive an error message indicating that the object already exists.

1. Edit QMF720.SDSQSAPE(DSQ1STGJ).
2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1STGJ PROC RGN='2048K',        Job-step region size
//     QMFTPRE='QMF720 ',           Prefix for QMF target libraries
//     DB2EXIT='DSN710 .SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710 .SDSNLOAD'   DB2 UDB for OS/390 program
                                    library name
```

3. Edit member DSQ1STGC.
4. Determine whether your user catalog is password protected. If it is, add the password clause to the STOGROUP statement in member DSQ1STGC.

```
CREATE STOGROUP PASSWORD(password)
```

5. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
CREATE STOGROUP DSQSGDEF
    VOLUMES (DSNVOL)      QMF  tables volume
    VCAT QMFDSN;          QMF  catalog alias
```

   The GRANT statements in this member allow *everyone* to create tables in the IVP table space. You can restrict this authority to certain users. You must include the installer if the installer is to run the IVP. If the program is run under the installer's authorization ID (the assumption), the installer automatically has the authority.
6. Submit QMF720.SDSQSAPE(DSQ1STGJ).

7. Check that you received a return code of 0. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Establishing the QMF sample tables

In the next two steps you establish the QMF sample tables.

"Step 11—Delete earlier sample tables" drops copies of sample tables created for a previous release of QMF.

"Step 12—Create the QMF sample tables" creates the QMF sample tables.

If you are migrating QMF from an earlier release, perform these steps. If you are installing QMF for the first time into a database, perform only "Step 12—Create the QMF sample tables".

Step 11—Delete earlier sample tables

The step deletes existing QMF sample tables from an earlier QMF version. It does not drop the six DB2 UDB for OS/390 views that were created in QMF Version 2.

1. Edit QMF720.SDSQSAPE(DSQ1EDSJ).
2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1EDSJ PROC RGN='2048K',       Job-step region size
//     QMFTPRE='QMF720 ',          Prefix for QMF  target libraries
//     DB2EXIT='DSN710.SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710.SDSNLOAD'   DB2 UDB for OS/390 program
                                   library name
```

3. Submit QMF720.SDSQSAPE(DSQ1EDSJ)
4. Check that you received a return code of 0. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and t rerun the job.

Step 12—Create the QMF sample tables

DSQ1EIVS creates the QMF sample tables. For more information about these tables, see Appendix B, "QMF Objects Residing in DB2", on page 729.

Tip: Sample tables are authorized to PUBLIC AT ALL LOCATIONS so that users can use three-part names to reference sample tables in another DB2 UDB for OS/390 subsystem.

QMF users at locations within the network are authorized to use all the sample tables created at the location into which you are installing QMF.

1. Edit QMF720.SDSQSAPE(DSQ1EIVS).
2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1EIVS PROC RGN='2048K',       Job-step region size
//     QMFTPRE='QMF720 ',          Prefix for QMF  target libraries
//     DB2EXIT='DSN710.SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710.SDSNLOAD'   DB2 UDB for OS/390 program
                                   library name
//     CDS='2',                    punctuation for decimal point
//     CDP='4'                       default is period (comma is CDS 6
//*                                                      CDP 7)
```

3. If you are migrating from another level of QMF, comment out job step 1.

```
//*STEP1   EXEC PGM=IKJEFT01,REGION=&RGN
```

4. Edit member QMF720.SDSQSAPE(DSQ1VSTC).

   This file creates a storage group, database, and table space for the sample tables.

5. Verify that the following parameters are correct:

```
CREATE STOGROUP DSQ1STBG
    VOLUMES (DSNVOL)   QMF  tables volume
    VCAT QMFDSN;       QMF  Catalog Alias in VCAT
```

6. Determine whether or not your user catalog is password protected. If it is, add the password clause to the STOGROUP statement in member DSQ1VSTC.

```
CREATE STOGROUP PASSWORD(password)
```

7. Submit job QMF720.SDSQSAPE(DSQ1EIVS).
8. Check that you received a return code of 0. Review SYSTERM for completion messages.

   Do not proceed if the return code is other than zero. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Step 13—Bind QMF packages

DSQ1BINJ binds the QMF packages into DB2 UDB for OS/390.

1. Edit QMF720.SDSQSAPE(DSQ1BINJ).

2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1BINJ PROC RGN='2048K',       Job-step region size
//     QMFTPRE='QMF720 ',          Prefix for QMF  target libraries
//     DB2EXIT='DSN710.SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//     DB2LOAD='DSN710.SDSNLOAD'   DB2 UDB for OS/390 program
                                   library name
```

3. Submit QMF720.SDSQSAPE(DSQ1BINJ).
4. Check for a return code of 4 or less.

   Do not proceed if the return code is higher than four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and if you need to rerun prior step(s), you will need to free QMF plans and packages first, then restart from step 8.

Step 14—Bind communications package to DB2 UDB for OS/390

Skip this step if you are not using the DB2 UDB for OS/390 communications package.

DSQ1BICD binds the DB2 UDB for OS/390 communications package to QMF.

1. Edit QMF720.SDSQSAPE(DSQ1BICD).
2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1BICD PROC RGN='2048K',       Job-step region size
//    QMFTPRE='QMF720 ',           Prefix for QMF  target libraries
//    DB2EXIT='DSN710.SDSNEXIT',   Exit DB2 UDB for OS/390  library name
//    DB2LOAD='DSN710.SDSNLOAD'    DB2 UDB for OS/390 program
                                   library name
```

3. Submit job QMF720.SDSQSAPE(DSQ1BICD).
4. Check for a return code of 4 or less.

   Do not proceed if the return code is higher than four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

Step 15—Bind QMF application plan to DB2 UDB for OS/390

Before you start QMF, you must bind it to DB2 UDB for OS/390. Before you bind QMF to DB2 UDB for OS/390, all of the DB2 UDB for OS/390 resources that QMF needs must be available to the authorization ID under which QMF is bound. At this point in the installation process, all of the essential DB2 UDB for OS/390 resources should be available.

DSQ1BINR is the job that does the binding; it binds the QMF application to DB2 UDB for OS/390 at the local DB2 UDB for OS/390.

1. Edit QMF720.SDSQSAPE(DSQ1BINR).
2. Verify that the installation parameters in the instream procedure of the job match your tailoring specifications. If they do not, return to "Step 2—Tailor the jobs" on page 66 and correct your installation parameters.

```
//DSQ1BINR PROC RGN='2048K',      Job-step region size
//    QMFTPRE='QMF720 ',          Prefix for QMF  target libraries
//    DB2EXIT='DSN710.SDSNEXIT',  Exit DB2 UDB for OS/390  library name
//    DB2LOAD='DSN710.SDSNLOAD'   DB2 UDB for OS/390 program
                                  library name
```

3. Submit job QMF720.SDSQSAPE(DSQ1BINR).
4. Check for a return code of 4 or less.

   Do not proceed if the return code is higher than four. Examine SYSTSPRT or SYSPRINT for error messages. Perform corrective actions and rerun the job.

At this point, you are ready to tailor QMF for TSO or CICS.

- For TSO, read Chapter 4, "Tailoring QMF for TSO", on page 47.
- For CICS, read Chapter 6, "Tailoring QMF for CICS", on page 69.

Chapter 4. Tailoring QMF for TSO

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

Step 16—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 QMF720.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 21.

   ```
   //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 QMF720.SDSQEXIT to the STEPLIB concatenation if needed. This only needs to be done if any exits reside in QMF720.SDSQEXIT.

      The sample includes the load libraries for ISPF, ISPF-PDF, QMF, DB2 UDB for OS/390, 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 OS/390 subsystem.

      If you plan to run concurrent versions of QMF with different plan IDs on the same DB2 UDB for OS/390 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 7 Release 2.0
      QMF720.SDSQLOAD
      Version 6
      QMF610.SDSQLOAD
      Version 3 Release 3.0
      QMF330.DSQLOAD
      Version 3 Release 2.0
      QMF320.DSQLOAD
      Version 3 Release 1.1
      QMF311.DSQLOAD
      Version 2 Release 2.4
      QMF240.DSQLOAD

```
//*********************************************************
//*        PROGRAM LOAD LIBRARIES                         *
//*********************************************************
//STEPLIB  DD  DSN=QMF720.SDSQEXIT,DISP=SHR      * QMF MODULES *
//         DD  DSN=QMF720.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=DSN710 .SDSNEXIT,DISP=SHR     * DB2 MODULES  *
//         DD  DSN=DSN710 .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.

```
//*********************************************************
//*        DATASETS USED BY TSO                           *
//*********************************************************
//SYSPROC  DD  DSN=SYS2.CLIST,DISP=SHR            * CLIST Library
//         DD  DSN=QMF720.SDSQCLTE,DISP=SHR
//SYSEXEC  DD  DSN=SYS2.EXEC,DISP=SHR
//         DD  DSN=QMF720.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.

```
//*********************************************************
//*        DATASETS USED BY ISPF                          *
//*********************************************************
//ISPPLIB  DD  DSN=QMF720.SDSQPLBE,DISP=SHR    * Panel libraries
//         DD  DSN=ISR.V4R1M0.ISRPLIB,DISP=SHR
//         DD  DSN=ISP.V4R1M0.ISPPLIB,DISP=SHR
//ISPMLIB  DD  DSN=QMF720.SDSQMLBE,DISP=SHR    * Message Libraries
//         DD  DSN=ISR.V4R1M0.ISRMLIB,DISP=SHR
//         DD  DSN=ISP.V4R1M0.ISPMLIB,DISP=SHR
//ISPSLIB  DD  DSN=QMF720.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=QMF720.DSQMAPE,DISP=SHR   * GDDM Map Group
//ADMCFORM  DD  DSN=QMF720.DSQCHART,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.

```
//*******************************************************
//*        DATASETS USED BY QMF                         *
//*******************************************************
//DSQPNLE   DD  DSN=QMF720DSQPNLE,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)
```

Data Extract (DXT) considerations

With administrative help, a user can start DXT dialogs. One method is to add JCL to your users' TSO logon procedure. A better way is to modify two CLISTs that IBM supplies with QMF.

Step 17—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 (QMF720).
   - With parameters:

     Choose the appropriate command for your type of install. If you are installing QMF into another DB2 UDB for OS/390 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.

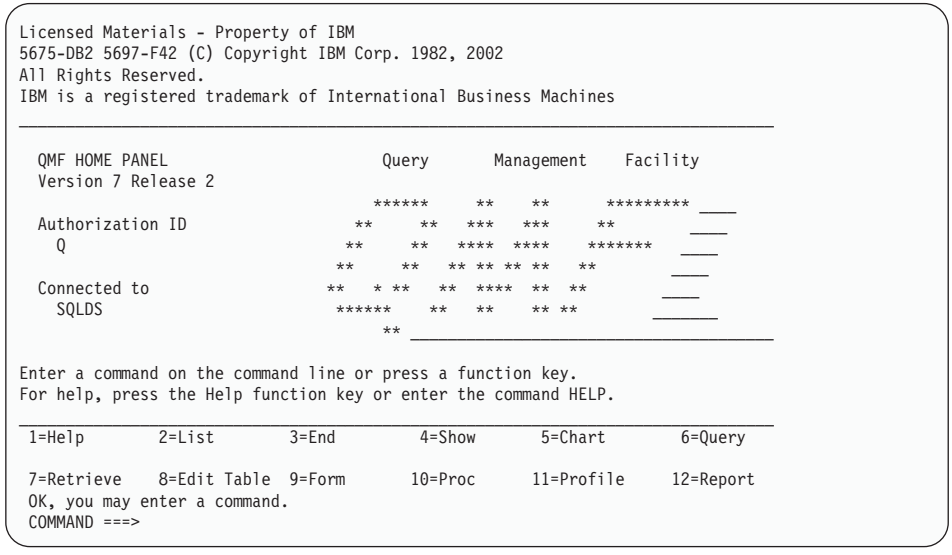


Figure 4. QMF Home Panel

- 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 53 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 QMF720.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 (see *Developing QMF Applications* for further details), or you can 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 'QMF720.SDSQLOAD(DSQQMFE)' 'DSQSSUBS=dbname,DSQSPLAN=planid,...'
  ```

- For requester installs:

```
CALL 'QMF720.SDSQLOAD(DSQQMFE)' 'DSQSSUBS=dbname,DSQSPLAN=planid,
      DSQSDBNM=<location>...'
```

For more information on starting QMF, see *Installing and Managing QMF for MVS*.

Step 18—Set up 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 "Step 35—Run the batch-mode IVP (optional)" on page 91. 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 9, "Testing Your QMF Install", on page 85.

Chapter 5. Providing Input Parameters

In this chapter you customize a CLIST with input parameters specific to your installation. Next, you run the job that updates the members with your parameter information.

Before performing the steps in this chapter, you must first install QMF in your OS/390 environment using SMP/E as documented in the *QMF Program Directory.*

Step 1—Provide QMF installation parameters

This step will loop through a series of QMF installation panels. The panels prompt you for the QMF and DB2 for OS/390 information that you described on the worksheets in Table 10 on page 27.

Before you start

Before starting this step, consider the following requirements:

1. To do this step, you must be in an active ISPF session.
2. From the command line of your ISPF session enter (`PANELID`) to turn your panel IDs on.
3. If you changed the default QMF target names (as originally specified in DSQ1EJAL), you must either alter the DSQ1EINS, DSQ1EIN1, and DSQ1EIN2 CLISTs, or skip to Chapter 3, “Submitting QMF Batch Install Jobs”.

   The tailoring portion of DSQ1EINS changes members in the SDSQSAPE and SDSQEXCE data sets. Make a backup copy of SDSQSAPE and SDSQEXCE before invoking DSQ1EINS. The backups can be deleted once the QMF installation is complete.
4. If you are performing one of the database-only installs (full, server, or requester), ensure that the QMF target libraries you are using for the installation cannot be accessed by users of other databases during installation.
5. If you are looping back through these procedures, be aware that the installation step “Converting CLIST records” on page 33 changes SDSQCLTE from FB to VB. Manually change it to FB to run DSQ1EINS.

Starting the installation panels

1. Enter the following:

   ```
   TSO EXEC 'prefix.SDSQCLTE(DSQ1EINS)' 'QMFPRE(prefix)'
   ```

where *prefix* is the QMF target library prefix from your worksheets.

This process generates one of the following:

- The Install QMF — Main menu, as shown in Figure 6, when you have completed and saved installation parameters.
- The Install QMF — Local DB2 UDB for OS/390 Parameters panel, as shown in Figure 7 on page 58, when there is no record of installation parameters.

```
DXYEIN00                 INSTALL QMF -- MAIN MENU
ISPF Command ===>




   Currently working on installation into DB2 UDB for OS/390 subsystem DSN

   You can now re-specify the install parameters, tailor the installation
   files, install QMF with the tailored files in foreground, quit and run
   the tailored install files in batch, or quit and return here later.


   ENTER CHOICE HERE        ===>               ("P" - INPUT PARAMETERS,
                                                "T" - TAILOR INSTALL FILES,
                                                "I" - INSTALL IN FOREGROUND,
                                                "X" - EXIT INSTALL DIALOGS)



   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 6. Install main menu

You will return to the main install menu, in a looping manner after successfully completing the input parameter. The main menu offers you four options:

P Installation parameters

T Tailor install files

Tailors all the required install data sets for QMF. This option lets you edit jobs to:

- Format the QMF GDDM maps and panel file
- Bind the QMF application plan to DB2 UDB for OS/390
- Create a SAVE DATA table space (optional)
- Delete the sample tables (migration installation only)
- Install the QMF sample tables
- Tailor the QMF plan ID and DB2 UDB for OS/390 subsystem name for the QMF callable interface (REXX EXEC DSQSCMDE)
- Set up the Installation Verification Procedures (IVP)

If you previously tailored files in SDSQSAPE and SDSQEXCE and want to keep them, back them up before you select the **T** option, because the input parameter procedures write over that information. This step is described further in "Step 2—Tailor the jobs" on page 66.

I Install in foreground (optional)

This option lets you submit your jobs in an online environment. You may also choose to submit your jobs manually, as discussed in Chapter 3, "Submitting QMF Batch Install Jobs", on page 31.

X Exit install dialogs- to end the series of panels

Also on this panel, you see the last-used DB2 UDB for OS/390 subsystem name. You can ignore that DB2 UDB for OS/390 name if you choose the P option, because the DB2 UDB for OS/390 name and other QMF install parameters might be overridden in the subsequent panels. Likewise, you can customize QMF install parameters for an additional DB2 UDB for OS/390 subsystem by ignoring the DB2 UDB for OS/390 subsystem name on the panel and proceeding to the next panel by entering P.

2. Choose the P option to obtain the first parameter input panel.

As you enter the information on each panel, QMF saves your input in the QMF720.SDSQCLTE library under your chosen database name.

If you leave this step before completing the last input parameter panel, your input is not saved. The last panel asks you for job card information used to tailor the installation. If you plan to install in the foreground, rather than through batch, you do not need to provide job card information; just enter x in the indicated spot on the panel.

After you supply the last installation parameter, you return to the main menu. If you want to review or modify the parameters, enter P and proceed through the input panels again. When you are satisfied with your installation parameters, continue with the next step. (If you prefer, you can leave the installation process at this point and return later; your installation parameters are saved.)

Specifying local DB2 UDB for OS/390 parameters

The panel displayed in Figure 7 on page 58 displays if you have not yet saved any install parameters. You also receive it if you choose the **P** option from the main menu.

Use the information from your worksheets to fill in the panel.

```
DXYEIN10                  INSTALL QMF -- LOCAL DB2 PARAMETERS
ISPF Command ===>


   LOCAL DB2 SUBSYSTEM ID    ===> DSN

   LOCAL DB2 RELEASE LEVEL   ===>     ("31" FOR V3R1, ETC)

   LOCAL DB2 EXIT LIBRARY    ===>

   LOCAL DB2 LOAD LIBRARY    ===>


   COMMUNICATIONS DATABASE(CDB) INSTALLED AT LOCAL DB2  ===>     ("Y","N")



   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 7. Local DB2 UDB for OS/390 parameters

The following options are available on this panel:

Local DB2 UDB for OS/390 subsystem ID
Specify the DB2 UDB for OS/390 subsystem ID where the QMF application plan is bound (required: default is DSN).

Local DB2 UDB for OS/390 release level
Specify the DB2 UDB for OS/390 release level of the local subsystem (required: no default).

Local DB2 UDB for OS/390 exit library
Specify the exit library for the local DB2 UDB for OS/390 subsystem (required: no default).

Local DB2 UDB for OS/390 load library
Specify the DB2 UDB for OS/390 load library for the local subsystem (required: no default).

Communications database (CDB) installed at local DB2 UDB for OS/390
Specify whether the DB2 UDB for OS/390 communications database is installed at the local DB2 UDB for OS/390 subsystem (required: no default).

Specifying the scope of database install

The panel displayed in Figure 8 on page 59 displays if you indicated that the communications database is installed at the local DB2 UDB for OS/390 subsystem on the previous panel.

```
DXYEIN12           INSTALL QMF -- SCOPE OF DATABASE INSTALL
ISPF Command ===>




   SCOPE OF DATABASE INSTALL       ===>   ("F" - full database,
                                             "R" - requester database only,
                                           "S" - server database only)




   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 8. Database install scope

Specify the scope of the database installation. For details about these options see "Road maps for the QMF installation process" on page 10.

If you are installing QMF Version 7.2 for the first time, select the full database install option.

Specifying QMF parameters for a local DB2 UDB for OS/390 subsystem

The panel displayed in Figure 9 on page 60 displays for full database and requester database installations.

```
DXYEIN11          INSTALL QMF -- QMF PARAMETERS AT LOCAL DB2
ISPF Command ===>



   CUSTOMIZE QMF RUNTIME LIBRARIES           ===> Y   ("Y" or "N")

     - Install QMF panels
     - Install QMF/GDDM map groups
     - Install QMF/GDDM sample charts forms
     - Make QMF REXX EXECs available
     - Make QMF CLISTs available



   QMF APPLICATION PLAN ID AT LOCAL DB2      ===> QMF720




   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 9. QMF parameters at local DB2 UDB for OS/390

The following options are available on this panel:

Customize QMF runtime libraries
Specify YES if the QMF runtime libraries require customizing. Customize these libraries only once per operating system (required: no default).

QMF application plan ID at local DB2 UDB for OS/390
Specify the QMF application plan name to be bound at the local DB2 UDB for OS/390 subsystem (required: no default).

Specifying remote server location

The panel displayed in Figure 10 on page 61 displays if you indicated *S* for server database, on the “Scope of database install” panel.

```
DXYEIN14              INSTALL QMF -- DB2 SERVER SYSTEM
ISPF Command ===>




DB2 SERVER LOCATION IN REMOTE DB2 SUBSYSTEM   ===> N  ("Y" OR "N")

   (If the DB2 server location is different from
    the requester location, the DB2 server is remote.)
```

Figure 10. DB2 UDB for OS/390 remote server panel

The following option is available on this panel.

DB2 UDB for OS/390 server location in remote DB2 UDB for OS/390 system

Specify whether the server database is different from the local DB2 UDB for OS/390 subsystem (required: no default).

Specifying DB2 UDB for OS/390 and QMF parameters

```
DXYEIN16            INSTALL QMF -- DB2 AND QMF PARAMETERS
ISPF Command ===>

   DB2 USER CATALOG                   ===>

   DB2 USER CATALOG PASSWORD          ===>


   QMF TABLESPACES CATALOG ALIAS      ===> QMFDSN

   QMF TABLESPACES CATALOG PASSWORD   ===>

   QMF TABLESPACES VOLUME             ===>  ( VOLUME SERIAL NUMBER
                                              OR "AST",
                                              AST stands for *)

   PREVIOUS QMF LEVEL                 ===>             ("V2R4","V3R1","V3R1M1",
                                                        "V3R2","V3R3","V6R1","NONE")

   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 11. DB2 UDB for OS/390 and QMF parameters

Fill in the following parameters:

DB2 UDB for OS/390 user catalog
Specify the ICF catalog that QMF install uses to create the QMF catalog alias (the VCAT name) (required: no default).

DB2 UDB for OS/390 user catalog password
Specify the password to access the DB2 UDB for OS/390 user catalog, which enables the QMF installation to create the QMF catalog alias in this user catalog (optional).

QMF tablespaces catalog alias
Specify the VCAT name for all the QMF table spaces. The VSAM data sets that associate with these QMF table spaces have the high-level qualifier of this alias value. If you are migrating from a previous level of QMF, use the same alias value as the previous release (required: no default).

QMF tablespaces catalog password
Specify the password for all the QMF control table spaces and index spaces created by the installation (optional).

QMF tablespaces volume
Specify a volume serial number where the QMF table spaces reside (required: no default required default required: no default).

Default punctuation
Specify the symbol for a decimal point in DB2 UDB for OS/390 (required; no default).

Previous QMF level
Specify the release level of QMF that you are migrating from (required: if you do not have any previous release level in the database, enter NONE).

Specifying remote server parameters

The panel displayed in Figure 12 on page 63 displays only when the server is different from the local DB2 UDB for OS/390 system.

```
DXYEIN15           INSTALL QMF -- REMOTE SERVER PARAMETERS
ISPF Command ===>

   DB2 SERVER LOCATION NAME                ===>
   DB2 SERVER ON A REMOTE OS/390 SYSTEM    ===>         ("Y" OR "N")
   DB2 USER CATALOG FOR SERVER    ===>
   DB2 USER CATALOG PASSWORD               ===>

   QMF TABLESPACES CATALOG ALIAS AT SERVER  ===> QMFDSN
   QMF TABLESPACES CATALOG PASSWORD         ===>
   QMF TABLESPACES VOLUME      ===>        ( VOLUME SERIAL NUMBER
                                             OR "AST",
                                             AST stands for *)

   DEFAULT PUNCTUATION AT SERVER            ===> .       ("," OR ".")
   PREVIOUS QMF LEVEL INSTALLED AT SERVER   ===>         (V2R4,V3R1,V3R3,
                                                          V3R1M1,V3R2,NONE)
   ROUTE XEQ JCL STATEMENT TO SERVER SYSTEM (REQUIRED IF SYSTEM IS REMOTE)
     FOR JES2, USE THE FORMAT: /*ROUTE XEQ <NODEID>.<USERID>
     FOR JES3, USE THE FORMAT: //*ROUTE XEQ <NODEID>.<USERID>
   ===>

   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 12. Remote server parameters

Fill in the following parameters:

DB2 UDB for OS/390 server location name
Specify the DB2 UDB for OS/390 location name for the remote server database (required: no default).

DB2 UDB for OS/390 server in another operating system
Specify whether the remote server database is in a different operating system than the requester database system (required: no default).

DB2 UDB for OS/390 user catalog for server
Specify the ICF catalog that QMF install uses to create the QMF catalog alias (QMF VCAT name) (required: no default).

DB2 UDB for OS/390 user catalog password
Specify the password to access the DB2 UDB for OS/390 user catalog so that the QMF installation creates a QMF catalog alias in this user catalog (optional).

QMF tablespaces catalog alias at server
Specify the VCAT name for all the QMF table spaces. The VSAM data sets that associate with these QMF table spaces have the high-level qualifier of this alias value. If you are migrating from a previous level of QMF, use the same alias value as the previous release (required: no default).

QMF tablespaces catalog password
Specify the password for all the QMF control table spaces and index spaces created by the installation (optional).

QMF tablespaces volume for server
Specify a volume serial number where the QMF table spaces reside (required: no default).

Default punctuation at server
Specify the symbol for a decimal point (required: no default).

Previous QMF level installed at server
Specify the release level of QMF that you are migrating from (required: if you do not have any previous release level in the database, enter NONE).

ROUTE XEQ JCL statement to server system
Specify the ROUTE JCL to send certain install jobs to the remote system for execution (required: if you have indicated that the server system is different from the requester system).

Specifying space parameters for QMF table spaces

The panel displayed in Figure 13 displays when the install scope is F (full database), or S (server database) install, and there is no previous QMF release level in the database.

```
DXYEIN17       INSTALL QMF -- QMF TABLESPACES SPACE PARAMETERS
ISPF Command ===>

   Specify the sizes (in 1K units) for the following tablespaces


      TABLESPACE FOR QMF
      CONTROL TABLE:        PRIMARY        SECONDARY
      ------------------    -------        ---------
      Q.OBJECT_DIRECTORY    ===> 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
     "SAVE DATA" TABLESPACE  ===> 100       ===> 20



   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 13. QMF table spaces space parameters

Specify the primary and secondary allocations for the QMF control table space. QMF uses these values to allocate all the VSAM files for these table spaces. Depending on the size of your installation, you might need to increase or decrease the default sizes to allow free space for expansion. Figure 13 shows the default sizes in 1K units.

Specifying parameters for QMF index spaces

The panel displayed in Figure 14 displays when the install scope is F (full database) or S (server database) install, and there is no previous QMF release level in the database.

```
DXYEIN18       INSTALL QMF -- QMF INDEXSPACES SPACE PARAMETERS
ISPF Command ===>

   Specify the sizes (in 1K units) for the following table indexes


      TABLE INDEX            PRIMARY        SECONDARY
      -----------            -------        ---------
      Q.OBJECT_DIRECTORYX    ===> 100       ===> 20
      Q.OBJECT_REMARKSX      ===> 100       ===> 20
      Q.OBJECT_OBJDATAX      ===> 100       ===> 20
      Q.PROFILEX             ===> 100       ===> 20
      Q.COMMAND_SYNONYMX     ===> 100       ===> 20









   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 14. QMF index spaces space parameters

The default sizes in 1K units are listed in Figure 14.

Specify the primary and secondary allocations for the QMF index spaces. QMF uses these values when allocating all the VSAM files for these table spaces. Depending on the size of your installation, you might need to increase or decrease the default sizes to allow free space for expansion.

Specifying the job card

The panel displayed in Figure 15 on page 66 is the last panel for the P option, installation parameters.

```
DXYEIN19                    INSTALL QMF -- JOBCARD
ISPF Command ===>

   Modify the Job cards below to represent your installation requirements.
   The "USER" and "PASSWORD" parameters must be specified in systems using
   RACF.  Since part of this install involves creating objects in DB2, you
   will need DB2 SYSADM authority.  Please see the "QMF Installation Guide
   for OS/390" for more detail.

   If you will be performing the installation in foreground rather than
   batch, and you DO NOT want the batch (JCL) files tailored, enter an
   'X' here:  ===>


   JOB CARD INFORMATION (used for batch (JCL) tailoring)
   ===> //QMFINSTL JOB (ACCT),NAME,
   ===> //          CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),
   ===> //          USER=Q,PASSWORD=Q
   ===> //*

   PRESS:  ENTER to continue    PF01 for help    PF03  to end
```

Figure 15. Job card

QMF uses this job card information to submit all of the remaining install jobs for your installation. When you complete this panel, you return to the *Install QMF — Main menu*. You can review your selections by choosing **P**, or continue on to tailoring the job.

Step 2—Tailor the jobs

Choose **T** on the main menu to tailor the jobs. This step updates the existing SDSQSAPE and SDSQEXCE members with the installation parameters settings you provided in "Step 1—Provide QMF installation parameters" on page 55.

During this step:

- A message tells you that the system is tailoring the JCL and copy files for the installation path you selected during "Step 1—Provide QMF installation parameters" on page 55.
- The QMF callable interface REXX EXEC QMF720.SDSQEXCE(DSQSCMDE) is modified for full database and requester installations to update the default values for the parameters: QMF plan ID and DB2 UDB for OS/390 subsystem name.

At the end of this step, you return to *Install QMF — Main Menu*. You can then proceed with the installation of QMF.

Attention: Do not manually edit any of the install JCL or control files in the QMF720.SDSQSAPE library, unless you are instructed to do so. The CLISTs that tailor these files depend on the sequence and format of the lines in these files.

If you choose to submit your job by batch, you are instructed to verify that the tailoring process worked correctly. Always return to the *Install QMF — Main Menu* for tailoring if the job results did not match your intentions.

Your next step is to choose to install QMF either in the foreground, or manually by batch. To choose a foreground install, select **I** on the *Install QMF — Main Menu* and follow the instructions in "Step 3—Install QMF in the foreground". To choose a manual install, select **X** on the *Install QMF — Main Menu* and follow the instructions in Chapter 3, "Submitting QMF Batch Install Jobs", on page 31.

When you choose **I**, a panel displays the installation options. When you have entered the information, a message tells you that the installation is proceeding.

Step 3—Install QMF in the foreground

When you select **I** on the *Install QMF — Main Menu* to submit the jobs in the foreground, they install under your current LOGON ID. Ensure your LOGON ID has SYSADM authority as described in "Database authorization ID Q" on page 10.

If you are installing QMF into a remote DB2 UDB for OS/390 server, do the following tasks before you run the foreground installation:

- If you are installing QMF for the first time:

  Use the full database installation option to install QMF into the local DB2 UDB for OS/390 subsystem.

  If you install QMF into a server that is not in the local DB2 UDB for OS/390 subsystem, submit the required job to allocate the VSAM data set before you invoke this CLIST in the foreground.

  For an initial install, submit the job QMF720.SDSQSAPE(DSQ1TBAJ).

- If you are migrating from QMF Version 2:

  If you are migrating from QMF Version 2 Releases 2, 3, or 4, use the full database installation to migrate QMF to the local DB2 UDB for OS/390.

  1. Issue the -STOP command in QMF720.SDSQSAPE(DSQ1SPDB) to stop the index space at the server.
  2. Submit QMF720.SDSQSAPE(DSQ1TBA1).
  3. Issue the -START command in QMF720.SDSQSAPE(DSQ1STDB) to start the index space at the server.

- If you are migrating from QMF Version 3 or Version 6:

  No action is required.

Foreground installation is now complete. Continue the installation by skipping to one of the following chapters:

- Chapter 4, “Tailoring QMF for TSO”, on page 47
- Chapter 6, “Tailoring QMF for CICS”, on page 69
- Chapter 7, “Tailoring QMF for Workstation Database Servers”, on page 75

Chapter 6. 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 OS/390 and GDDM for CICS. For more details, see the *GDDM Installation and System Management* and the *DB2 UDB for OS390 Administration Guide*.

Step 19—Describe QMF to DB2 UDB for OS/390 in CICS

This step ensures that you complete all the prerequisite steps before you tailor QMF for CICS.

1. Ensure that you install the DB2 UDB for OS/390-to-CICS connection and the DB2 UDB for OS/390 attachment facility for CICS.

   QMF uses the CICS/DB2 attachment facility to access DB2 UDB for OS/390 data in the CICS environment. QMF-specific information for these products is described in the *DB2 UDB for OS390 Administration Guide*.
2. Verify that the plan ID and authorization IDs for a QMF transaction ID are in the resource control table (RCT).

   Users invoking a QMF transaction operate under the authorization of the associated RCT entry. (For a sample RCT, see member DSQ1ERCT in the QMF sample library QMF720.SDSQSAPE.)

   If your RCT includes RACF information, the authorization ID must be a valid RACF ID.
3. Regenerate your RCT as described in *DB2 UDB for OS390 Administration Guide*

   For a complete description of the resource control table, see the *DB2 UDB for OS390 Administration Guide*.

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

Step 20—Link-edit QMF with DFHEAI and DFHEAI0

This step uses two jobs, DSQ1ELNK and DSQ1EGLK, to link-edit QMF with the CICS interface modules, DFHEAI and DFHEAIO. Because QMF uses the CICS command level application programming interface when operating under CICS, you must link-edit before you can run any QMF programs.

Link-edit QMF with CICS command interface modules

DSQ1ELNK link-edits QMF with the CICS command interface modules, DFHEAI and DFHEA10, located in the LOADLIB data set generated by CICS.

Important: To include CICS interface modules DFHEAI and DFHEAI0, you must run this step each time you apply QMF service.

1. Edit QMF720.QMFSAMPE(DSQ1ELNK).
2. Verify that the installation parameters in the instream procedure of the job, and the job steps, match your tailoring specifications.

```
//DSQ1ELNK PROC REG=4096K,        Job Step Region
//      QMFTPRE='QMF720 ',         DSN Prefix for QMF product
//      CLOAD='CICS.LOADLIB',     Name of CICS LOADLIB
//      OUTC='*                   Print SYSOUT class
```

3. Submit QMF720.QMFSAMPE(DSQ1ELNK).
4. Check for a return code of 0. If the return code is not 0, correct the problem and rerun DSQ1ELNK.

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

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

1. Edit QMF720.QMFSAMPE(DSQ1EGLK).
2. Verify that the installation parameters in the instream procedure of the job, and the job steps, match your tailoring specifications.

```
//DSQ1EGLK PROC SUFFIX=1$,        CICS ASM Translator suffix
//      QMFTPRE='QMF720 ',         DSN Prefix for QMF product
//      CMACS='CICS.MACLIB',      Name of CICS MACLIB
//      CLOAD='CICS.LOADLIB',     Name of CICS LOADLIB
//      A=,                        A=A for CICS Aligned MAP
//      ASMBLR=IEV90,             Assembler Program Name
//      REG=4096K,                Job step region
//      OUTC='*',                 Print SYSOUT class
//      WORK='SYSDA'              Work unit
```

3. Submit QMF720.QMFSAMPE(DSQ1EGLK).
4. Check for a return code of 0 on all jobs 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.

Step 21—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 QMF720.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='QMF720 ',   QMF prefix name for target libraries
//         GDDMADM='GDDM.ADMF' GDDM ADMF data set name
```

3. Submit QMF720.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 7.2 from a previous QMF release, skip this step.

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

1. Edit QMF720.SDSQSAPE(DSQ1BFRM).
2. Locate the installation parameters in the instream procedure of the job and ensure that they match your specifications.

```
//DSQ1BFRM PROC QMFTPRE='QMF720 ',     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(QMF720.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 QMF720.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.

Step 22—Update CICS control tables (CICS version 3 or later)

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 QMF720.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 DSQ1CSCS 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 RCT and DCT. For other RCT considerations, see "Step 19—Describe QMF to DB2 UDB for OS/390 in CICS" on page 69.

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 QMF720.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='QMF720 ',         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.

   Note: For CICS V4 and later you can ignore the following error:

   ```
   E 'RESSECNUM' is not valid and is ignored
   ```

   from the DEFINE FILE(DSQPNLE) statement or the DEFINE FILE(DSQUCFRM) statement. Any other errors should be corrected.

Step 23—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.

Step 24—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 link-edited RCT is accessible to OS/390 through the normal library search order (STEPLIB, JOBLIB, link library).

```
//STEPLIB  DD DSN=CICS.SDFHAUTH,DISP=SHR
//         DD DSN=DSN710.SDSNEXIT,DISP=SHR
//         DD DSN=DSN710.SDSNLOAD,DISP=SHR
```

In this example, DFHSIP, which loads from CICS.LOADLIB1, must receive control in an authorized state. You must individually APF-authorize each concatenated library.

DSN.SDSNLOAD is the library containing the link-edited RCT, it must also be authorized.

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 OS/390 modules in the CICS module load library list, DFHRPL.

```
//DFHRPL  DD  ...
//        DD  DSN=QMF720.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 OS/390 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=QMF720.DSQPNLE,DISP=SHR   QMF Panel File
//DSQDEBUG  DD DSN=QMF720.DSQDEBUG,DISP=SHR  Trace and Error Messages
//DSQUCFRM  DD DSN=QMF720.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 9, “Testing Your QMF Install”, on page 85.

Chapter 7. Tailoring QMF for Workstation Database Servers

In this chapter all of the following DB2 products are referred to collectively as the DB2 DRDA AS. When it is necessary, a more specific reference is made to one of the following products:

- DB2 Universal Database™ Version 5 (UDB for AIX, DB2 for NT, ...)
- DB2 Common Server Version 2.1 (DB2 for AIX, DB2 for NT, ...)
- DB2 Parallel Edition Version 1.2
- DataJoiner® Version 1.2

QMF support for a DB2 DRDA AS is optional. You only need to perform the steps described in this chapter 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 OS/390, you need to make the following preparations for DB2 Common Server, DB2 Parallel Edition, or DataJoiner:

- Create an installation ID on the platform of the DB2 DRDA AS and make it a member of the SYSADM GROUP.
- Create the database on the platform of 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 platform of 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" <sysadm-id> "using" <password>

        "db2 get authorizations"
  ```
- (Optional) Grant additional administrative authorities to groups, users, or PUBLIC, as needed. If you install QMF into DB2 for VM or VSE server from OS/390, you must create public and private dbspaces. QMF needs some of the public dbspaces for tables, queries, procedures, forms, and data.

The following steps apply to QMF migration and to first-time QMF installations.

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 support.

Step 25—Bind QMF install programs to DB2 DRDA AS

This step binds programs DSQDBSQL and DSQDBINS to the DB2 DRDA AS. The application plan associated with these packages is DSQSI720.

1. Edit QMF720.SDSQSAPE(DSQ1BDJ1).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQBIND PROC RGN='2048K',    Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change <ssid> to your DB2 for OS/390 subsystem ID.
4. Change <location> to the location name of the DB2 DRDA AS database application server as defined in the DB2 for OS/390 communications database.
5. (Optional) Review the Comments in the JOB for further tailoring opportunities.
6. Submit job QMF720.SDSQSAPE(DSQ1BDJ1).
7. Check Procstep BIND for a return code of 0. Examine SYSTSPRT for error messages. Do not proceed if the return code is other than zero. Perform corrective actions if required and rerun the job.

Step 26—Create QMF control tables in a DB2 DRDA AS

This step creates the QMF control tables in DB2 DRDA AS.

1. Edit QMF720.SDSQSAPE(DSQ1EDJ2).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQEXSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMf720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change <ssid> to your DB2 for OS/390 subsystem ID.
4. (Optional) Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF720.SDSQSAPE(DSQ1EDJ2).

6. Check Stepname DSQCTBL 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 27—Bind QMF application programs to a DB2 DRDA AS

This step binds QMF application programs to DB2 DRDA AS. After successful completion of this step, QMF Version 7.2 can connect to the DB2 DRDA AS.

1. Edit QMF720.SDSQSAPE(DSQ1BPKG).
2. Verify and change, if necessary, the default values for the parameters in the job's instream procedure:

```
//DSQBIND PROC RGN='2048K',    Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change <ssid> to your DB2 UDB for OS/390 subsystem ID.
4. Change to your DB2 UDB for OS/390 application requester local subsystem ID.
5. Change <location> to the location name of the DB2 DRDA AS database application server as defined in the DB2 UDB for OS/390communications database.
6. (Optional) Review the Comments in the JOB for further tailoring opportunities.
7. Submit job QMF720.SDSQSAPE(DSQ1BPKG).
8. A return code of 0 or 4 indicates a successful run of this job. Review SYSTSPRT, SYSTERM and SYSPRINT outputs for clues to errors for return codes greater than 4. Perform corrective actions and rerun the job.

Step 28—Create QMF sample tables in a DB2 DRDA AS

This step creates the QMF sample tables in DB2 DRDA AS.

1. Edit QMF720.SDSQSAPE(DSQ1EDJ4).
2. Verify and change, if necessary, the default values for the installation parameters in the both of the job's instream procedures:

```
//DSQEXSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name

//DSQINSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change <ssid> to your DB2 UDB for OS/390 subsystem ID.
4. (Optional) Review the comments in the job for further tailoring opportunities.
5. Submit job QMF720.SDSQSAPE(DSQ1EDJ4).
6. Check Stepname DSQSINS 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 then rerun this job.

Deleting QMF from a DB2 DRDA AS

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

Deleting QMF

This step should be run 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 QMF720.SDSQSAPE(DSQ1EDX1).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQEXSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change <ssid> to your DB2 for OS/390 subsystem ID.
4. (Optional) Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF720.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

This step should be run 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 QMF720.SDSQSAPE(DSQ1EDX2).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQEXSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change <ssid> to your DB2 UDB for OS/390 subsystem ID.
4. (Optional) Review the Comments in the JOB for further tailoring opportunities.
5. Submit job QMF720.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.

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:

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

where <ssid> is your DB2 UDB for OS/390 subsystem ID and <location> is your DB2 DRDA AS location name.

You are ready to continue on to Chapter 9, “Testing Your QMF Install”, on page 85.

Chapter 8. Tailoring QMF for DB2 Universal Database for iSeries® Servers

Beginning with Version 7 Release 1, QMF supports connectivity from a QMF application requester to a DB2 UDB for iSeries Version 4 Release 4 (or higher) server. This support is optional. You need to perform the steps in this chapter only if you intend to connect QMF to a DB2 UDB for iSeries Version 4 Release 4 (or later) server. Before you install QMF on a DB2 UDB for iSeries server, you need to make the following preparations:

- From the DB2 UDB for iSeries Query Manager, execute the following SQL on the server: `CREATE COLLECTION Q` using a user ID that has administrative authority. This must be the security officer or a user ID that has *ALLOBJ authority.
- Ensure that users of QMF have *USE authority for Q *LIB.

The following steps apply to a first time QMF installation. You can rerun these steps again as necessary to correct errors.

Step 29—Bind QMF install programs to DB2 UDB for iSeries

This step binds programs DSQDBSQL and DSQDBINS to DB2 UDB for iSeries. The application plan associated with these packages is DSQSI720.

1. Edit QMF720.SDSQSAPE(DSQ1BAS1).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQBIND PROC RGN='2048K',    Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change to your DB2 for OS/390 subsystem ID.
4. Change to the location name of the DB2 UDB for iSeries database server as defined in the DB2 for OS/390 communications database.
5. Optional: Review the comments in the job for further tailoring opportunities.
6. Submit job QMF720.SDSQSAPE(DSQ1BAS1).
7. Check the return code of the job. Examine SYSTSPRT for error messages. Do not proceed if the return code is other than 0. Perform corrective actions, if required, and run the job again.

Step 30—Create QMF control tables in a DB2 UDB for iSeries server

This step creates the QMF control tables in DB2 UDB for iSeries server.

1. Edit QMF720.SDSQSAPE(DSQ1EAS2).
2. Verify and change, if necessary, the default values for the installation parameters in the job's instream procedure:

```
//DSQEXSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change to your DB2 for OS/390 subsystem ID.
4. Optional: Review the comments in the job to determine whether further tailoring is needed.
5. Submit job QMF720.SDSQSAPE(DSQ1EAS2).
6. Check Stepname DSQCTBL for a return code of 0 or 4. Review the complete job output for error messages.

   Do not proceed if the return code is other than 0 or 4. After reviewing the complete job output, perform corrective actions and run the job again.

Step 31—Bind QMF application programs to a DB2 UDB for iSeries server

This step binds QMF application programs to DB2 UDB for iSeries. After successful completion of this step, QMF Version 7.2 can connect to the DB2 UDB for iSeries server.

1. Edit QMF720.SDSQSAPE(DSQ1BPKG).
2. Verify and change, if necessary, the default values for the parameters in the job's instream procedure:

```
//DSQBIND PROC RGN='2048K',    Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change to your DB2 Universal Database for OS/390 subsystem ID.
4. Change to your DB2 Universal Database for OS/390 application requester local subsystem ID.
5. Change <location> to the location name of the DB2 UDB for iSeries database application server as defined in the DB2 Universal Database for OS/390 communications database.
6. Optional: Review the Comments in the job to determine whether further tailoring is needed.
7. Submit job QMF720.SDSQSAPE(DSQ1BPKG).
8. A return code of 0 or 4 indicates a successful run of this job. Review SYSTSPRT, SYSTERM and SYSPRINT outputs for clues to errors for return codes greater than 4. Perform corrective actions and run the job again.

Step 32—Create QMF sample tables in a DB2 UDB for iSeries server

This step creates the QMF sample tables in DB2 UDB for iSeries.

1. Edit QMF720.SDSQSAPE(DSQ1EAS4).
2. Verify and change, if necessary, the default values for the installation parameters in the both of the job's instream procedures:

```
//DSQEXSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name

//DSQINSQL PROC RGN='2048K',   Job-step region size
//  QMFTPRE='QMF720',          Prefix for QMF target libs
//  DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//  DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change to your DB2 Universal Database for OS/390 subsystem ID.
4. Optional: Review the comments in the job to determine whether further tailoring is needed.
5. Submit job QMF720.SDSQSAPE(DSQ1EAS4).
6. Check Stepname DSQSINS 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 run the job again.

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 UDB for iSeries.

Specify the following:

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

where <ssid> is your DB2 Universal Database for OS/390 subsystem ID and <location> is your DB2 UDB for iSeries location name.

Chapter 9. Testing Your QMF Install

This chapter describes the final steps in the install process.

The chapter includes the steps:
- Step 33 (for TSO) - Run the IVP
- Step 33 (for CICS) - Run the IVP
- Step 34 - Install the QMF application queries and application objects (TSO)
- Step 35 - Run the batch-mode IVP (optional)
- Step 36 - Clean up after install
- Step 37 - Accept the permanent libraries
- Step 38 - Clean up security

Step 33 (for TSO)—run the IVP

This step leads you through the final testing of QMF, called the installation verification procedure (IVP). To test QMF for OS/390 installation, you need to start QMF and issue a few QMF commands. 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 OS/390 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 OS/390 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 TABLESPACE 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 in "Step 10— Create a table space for the QMF IVP" on page 41, 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 "Step 17—Start QMF" on page 51.

   The QMF Home panel displays.

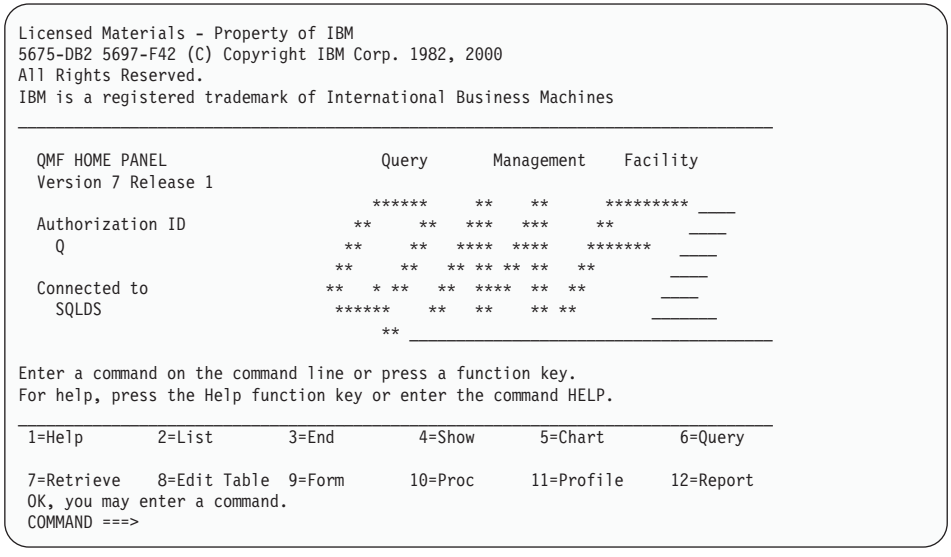


Figure 16. QMF Home panel

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 for OS/390 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 723 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.

The installation verification for interactive mode is now complete.

Step 33 (for CICS)—Run the IVP

This step leads you through the final testing of QMF, called the installation verification procedure (IVP). To test that you have installed QMF for MVS/CICS properly, you need to start QMF and issue a few QMF commands. 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 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 “Step 22—Update CICS control tables (CICS version 3 or later)” on page 72 for details.

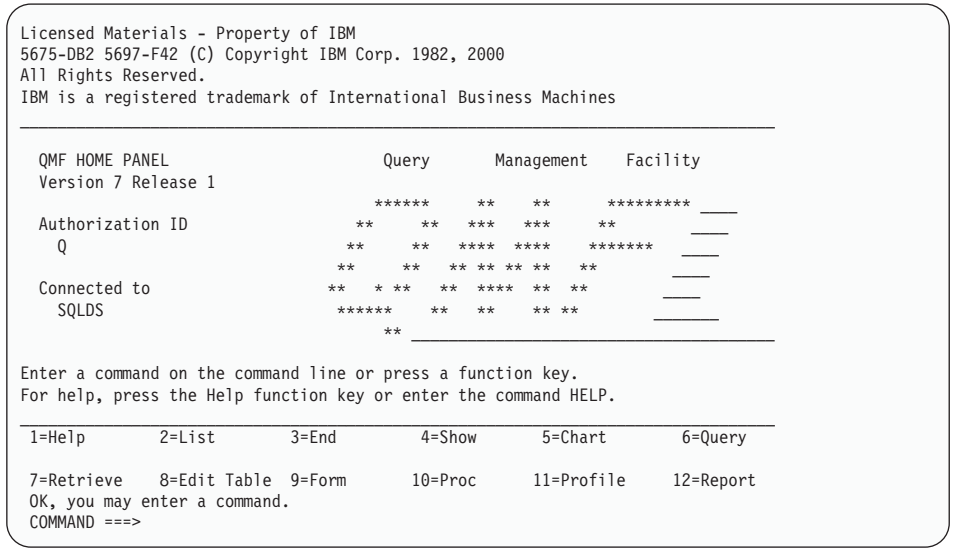
Start and test QMF

This procedure starts the QMF for MVS/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 723.

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.



4. Verify 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, 1998
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 7                                          |
|     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 is empty.

Step 34—Install the QMF application queries and application objects (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 OS/390 subsystem, skip this step.

   a. Begin a QMF session.
   b. Connect to the Workstation Database Server or DB2 for OS/390 server in which you just installed QMF.
   c. Enter the following command from within QMF:

      ```
      IMPORT PROC FROM 'QMF720.SDSQSAPE(DSQ1ESQD)'
      ```

      where *QMF720* 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 'QMF720.SDSQSAPE(DSQ1ESQI)'
   ```

where *QMF720* 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.

Step 35—Run the batch-mode IVP (optional)

Skip this step if your installation doesn't use batch-mode QMF.

This step tests batch-mode IVP by running the batch-mode job that you created in "Step 18—Set up QMF batch job to run batch IVP (optional)" on page 54. 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 QMF720 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.

Step 36—Clean up after install

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 7.2 and the release are in the same DB2 UDB for OS/390 subsystem.
- QMF Version 7.2 and a previous release are in different DB2 UDB for OS/390 subsystems

  This step removes the previous release when QMF Version 7.2 and the release are in different DB2 UDB for OS/390 subsystems.

After running either of these two substeps, you can delete the libraries of the previous QMF release. Table 11 on page 93 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 7.2 data set.

Table 11. Libraries to be deleted from earlier QMF releases

V2R4 data sets	V3RxMy data sets	V6R1 data sets
QMF240.DSQOBJ	QMF3xy.ADMFE	QMF610.SDSQCLTE
QMF240.DSQMACE	QMF3xy.CICS.DFHTEMP	QMF610.SDSQEXCE
QMF240.DSQPMSE	QMF3xy.DSQPMSE	QMF610.SDSQMLBE
QMF240.DSQDBRMD	QMF3xy.DSQDBRMD	QMF610.SDSQPLBE
QMF240.DSQSAMPE	QMF3xy.DSQSAMPE	QMF610.SDSQSAPE
QMF240.DSQMAPE	QMF3xy.DSQMAPE	QMF610.SDSQSLBE
QMF240.DSQCLSTE	QMF3xy.DSQCLSTE	QMF610.SDSQUSRE
QMF240.DSQEXECE	QMF3xy.DSQEXECE	QMF610.SDSQLOAD
QMF240.DSQUSERE	QMF3xy.DSQUSERE	QMF610.SDSQDBRM
QMF240.DSQPLIBE	QMF3xy.DSQPLIBE	QMF610.DSQMAPE
QMF240.DSQSLIBE	QMF3xy.DSQSLIBE	QMF610.DSQCHART
QMF240.DSQMLIBE	QMF3xy.DSQMLIBE	QMF610.DSQPVARE
QMF240.DSQLOAD	QMF3xy.DSQLOAD	QMF610.DSQPNLE
QMF240.DSQDBRM	QMF3xy.DSQDBRM	QMF610.ADSQOBJ
QMF240.DSQTLIBE	QMF3xy.DSQTLIBE	QMF610.ADSQDBMD
QMF240.DSQCHART	QMF3xy.DSQCHART	QMF610.ADSQMACE
	QMF3xy.DSQMACE	QMF610.ADSQPMSE
	QMF3xy.DSQOBJ	QMF610.DSQDEBUG
	QMF3xy.DSQPNLE	
	QMF3xy.DSQPVARE	
	QMF3xy.DSQUCFRM	

Freeing an earlier application plan

Run this step only when QMF Version 7.2 and the previous release are on the same DB2 UDB for OS/390 subsystem.

1. Edit QMF720.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='QMF720',             QMF prefix
//             DB2EXIT='DSN710.SDSNEXIT',    DB2 UDB for OS/390
//                                             exit library
//             DB2LOAD='DSN710.SDSNLOAD'     DB2 UDB for OS/390
 program library
```

4. Edit QMF720.SDSQSAPE(DSQ1DEL1).
5. Replace DSN with the name of the DB2 UDB for OS/390 subsystem, and replace QMF720 with the name of the application plan of the previous release.

```
DSN SYSTEM(DSN)
FREE PLAN(QMF720)
```

Table 12. QMF release defaults

Previous Release	Default
QMF Version 6.1	QMF610
QMF Version 3.3	QMF330
QMF Version 3.2	QMF320
QMF Version 3.1.1	QMF311
QMF Version 3.1	QMF310

6. Submit the job QMF720.SDSQSAPE(DSQ1JFPL).

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

QMF Version 7.2 and a previous release are in different DB2 UDB for OS/390 subsystems

Run this step only if QMF Version 7.2 and the earlier release are on different DB2 UDB for OS/390 subsystems. The step frees the earlier application plan and drops various DB2 UDB for OS/390 entities that belong to the earlier QMF release.

Attention: This job removes all traces of QMF from the DB2 UDB for OS/390 subsystem and should be run only if the current release of QMF does not exist in the DB2 UDB for OS/390 subsystem.

1. Edit QMF720.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='QMF720',             QMF prefix
// DB2EXIT='DSN710.SDSNEXIT',    DB2 UDB for OS/390 exit library
// DB2LOAD='DSN710.SDSNLOAD'     DB2 UDB for OS/390 program library
```

4. Edit member QMF720.SDSQSAPE(DSQ1DEL1).
5. Replace DSN with the name of the DB2 UDB for OS/390 subsystem and replace QMF720 with the name of the application plan of the previous release.

```
DSN SYSTEM(DSN)
FREE PLAN(QMF720)
```

Table 13. QMF release defaults

Previous Release	Default
QMF Version 6.1	QMF610
QMF Version 3.3	QMF330
QMF Version 3.2	QMF320
QMF Version 3.1.1	QMF311
QMF Version 3.1	QMF310

6. Edit member QMF720.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 ("Step 17" on page 41), delete the following statement:

   ```
   DROP STOGROUP DSQSGDEF
   ```

   QMF720

7. Edit member QMF720.SDSQSAPE(DSQ1DEL13)

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

8. Submit job QMF720.SDSQSAPE(DSQ1DELA).

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

If you are performing a requester or server database installation, go to "Step 38—Clean up security".

Step 37—Accept the permanent libraries

DSQ1EJAC runs the SMP/E ACCEPT job, which makes the libraries permanent.

Attention: If you have QMF Version 7.2 installed in the same DB2 UDB for OS/390 subsystem as a previous release, do not accept the permanent libraries until your local acceptance testing is complete. You can accept the libraries at any time after successful execution of the IVP.

Step 38—Clean up security

The JCL currently contains a valid user ID and password, which creates a security exposure. Correct this exposure as soon as possible. One possible solution is to edit the JCL and blank out the password value.

The installation control files contain passwords for the DB2 UDB for OS/390 catalog as well as for all the QMF control table spaces. Delete these passwords or restrict access to them. They are in QMF720.SDSQCLTE(xxxxINST), where "xxxx" is the DB2 UDB for OS/390 subsystem ID.

Chapter 10. Planning and Installing a QMF NLF

A QMF NLF 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 18 on page 100 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, CICS, and CMS environments. A unique row is added for each NLF that you install.

The NLF must be installed in each DB2 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.

Note: You must accept the QMF Version 7.2 base product just before installing a QMFNLF. It is assumed that QMF is sharing the DB2 SMP/E data sets.

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, the distribution libraries, the target libraries, and the 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 SMP/E storage" on page 23 for SMP/E requirements for installing base QMF. QMF and its features are added to the SMP/E data sets.

SMP/E data sets for QMF NLF

Additional estimated DASD space required (in cylinders) for the SMP/E data sets is shown in Table 14.

Table 14. Additional DASD space for SMP/E data sets (cylinders)

DDNAME	3380	3390	9345
SMPSCDS	1	1	1
SMPLOG	1	1	1
SMPMTS	1	1	1
SMPPTS	1	1	1
SMPSTS	1	1	1
SMPCSI	8	8	8

Distribution libraries for QMF NLF

The QMF Version 7.2 distribution libraries for the NLF are:

- QMF720.ADSQMACn, which contains QMF NLF installation procedures, IVP, sample queries, and QMF procedures.
- QMF720.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 15.

Table 15. Additional DASD space for QMF NLF distribution libraries (cylinders)

DSNAME	Content	3380	3390	9345
QMF720.ADSQMACn	QMF NLF install procs	15	13	15
QMF720.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 Table 16 on page 99.

Table 16. Additional DASD space for QMF NLF target libraries (cylinders)

DSNAME	3380	3390	9345
QMF720.SDSQSAPn	17	15	17
QMF720.SDSQPLBn	1	1	1
QMF720.SDSQCLTn	2	1	2
QMF720.SDSQMLBn	1	1	1
QMF720.SDSQEXCn	1	1	1
QMF720.SDSQUSRn			

QMF NLF user data sets

Estimated DASD space required (in cylinders) for the QMF NLF user data sets is shown in Table 17.

Table 17. DASD space for QMF NLF user data sets (cylinders)

DSNAME	Content	3380	3390	9345
QMF720.DSQMAPn	GDDM map group files	1	1	1
QMF720.DSQPVARn	QMF message help panels (expanded in sequential format)	6	6	N/A
QMF720.DSQPNLn	QMF message help panels (expanded in VSAM format)	10	9	N/A

IBM software distribution (ISD) tape

To install a QMF NLF, you first read in information from the IBM ISD tape. The tape contains the following:

- SMP/E control statements
- JCLIN for QMF 7 NLF
- JCL for installation-verification procedures
- Programs in load module format
- Panels and other items used by QMF Version 7.2 NLF

The ISD tape has an SMP/E (RELFILE) format. The format is fully 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 18 on page 100.

Table 18. Language ID and FMID

National Language Feature	Language ID	QMF 7 FMID
U/C English	U	JSQ7751
Danish	Q	JSQ7755
French	F	JSQ7756
German	D	JSQ7757
Italian	I	JSQ7758
Japanese Kanji	K	JSQ7759
Korean Hangeul	H	JSQ775A
Brazil Portuguese	P	JSQ775B
Spanish	S	JSQ775C
Swedish	V	JSQ775D
Swiss French	Y	JSQ775E
Swiss German	Z	JSQ775F
Canadian French	C	JSQ775G

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. The following figures show the steps required to install QMF 7 in a DB2 subsystem with or without an earlier QMF NLF release.

The NLF requires the use of the QMF Version 7.2 sample library, QMF720.SDSQSAPE, and the load module library, QMF720.SDSQLOAD.

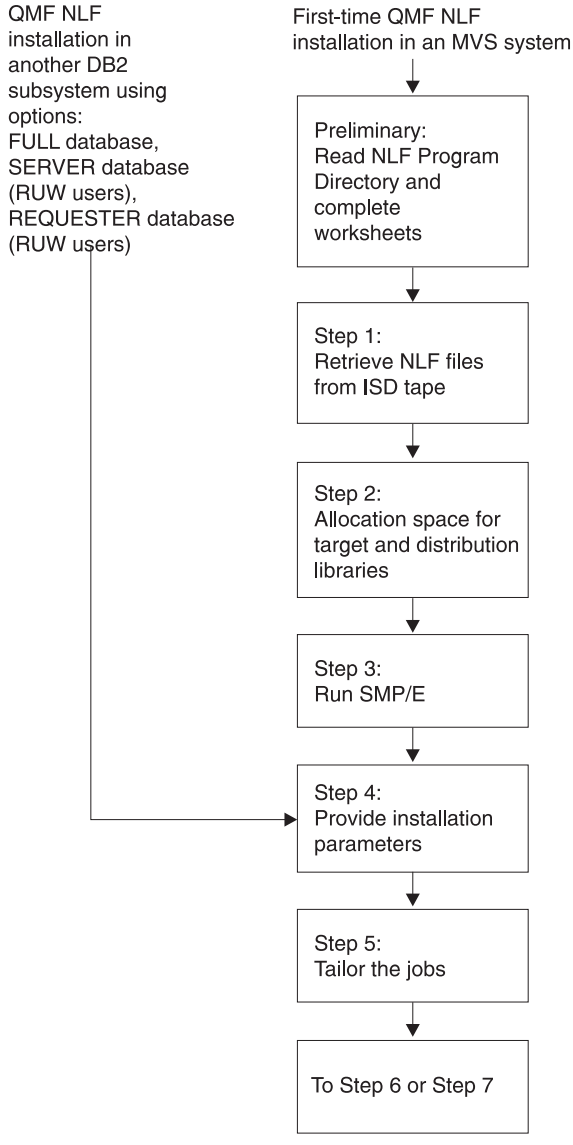


Figure 17. Installation steps for QMF NLF -- Part 1

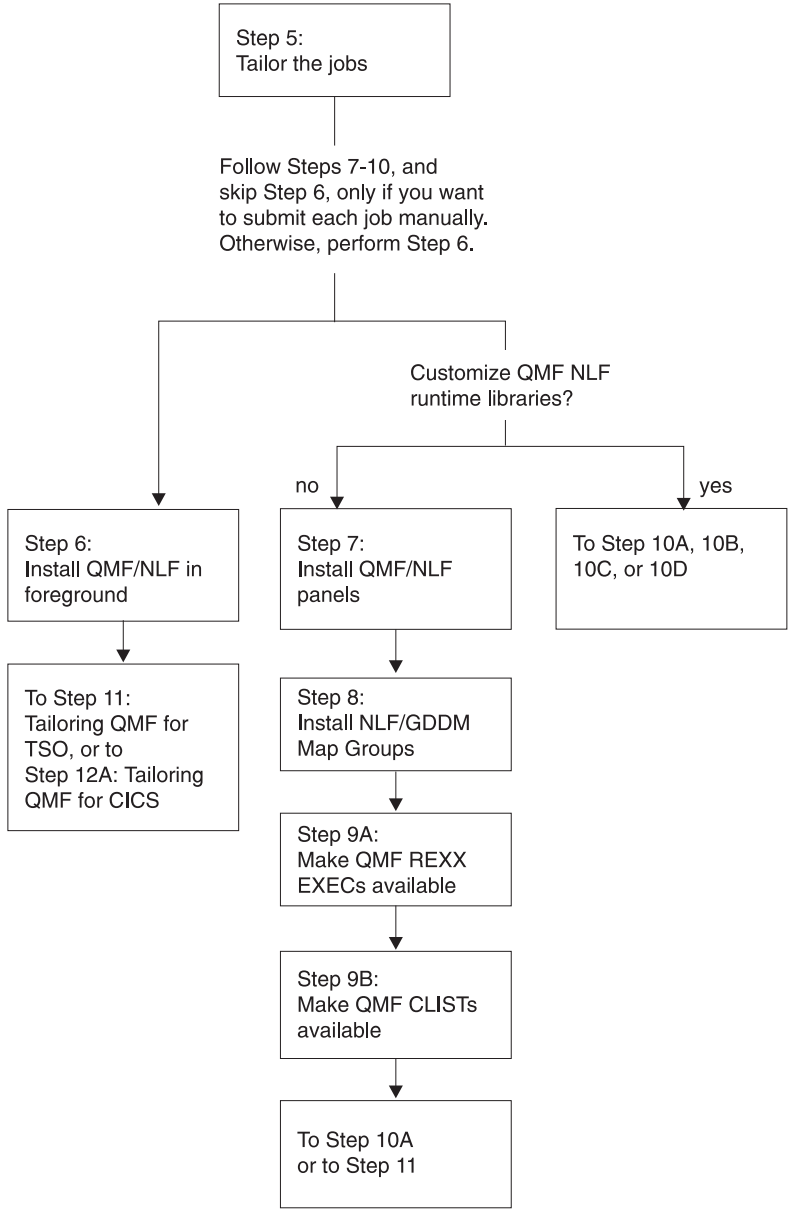


Figure 18. Installation Steps for QMF NLF -- Part 2

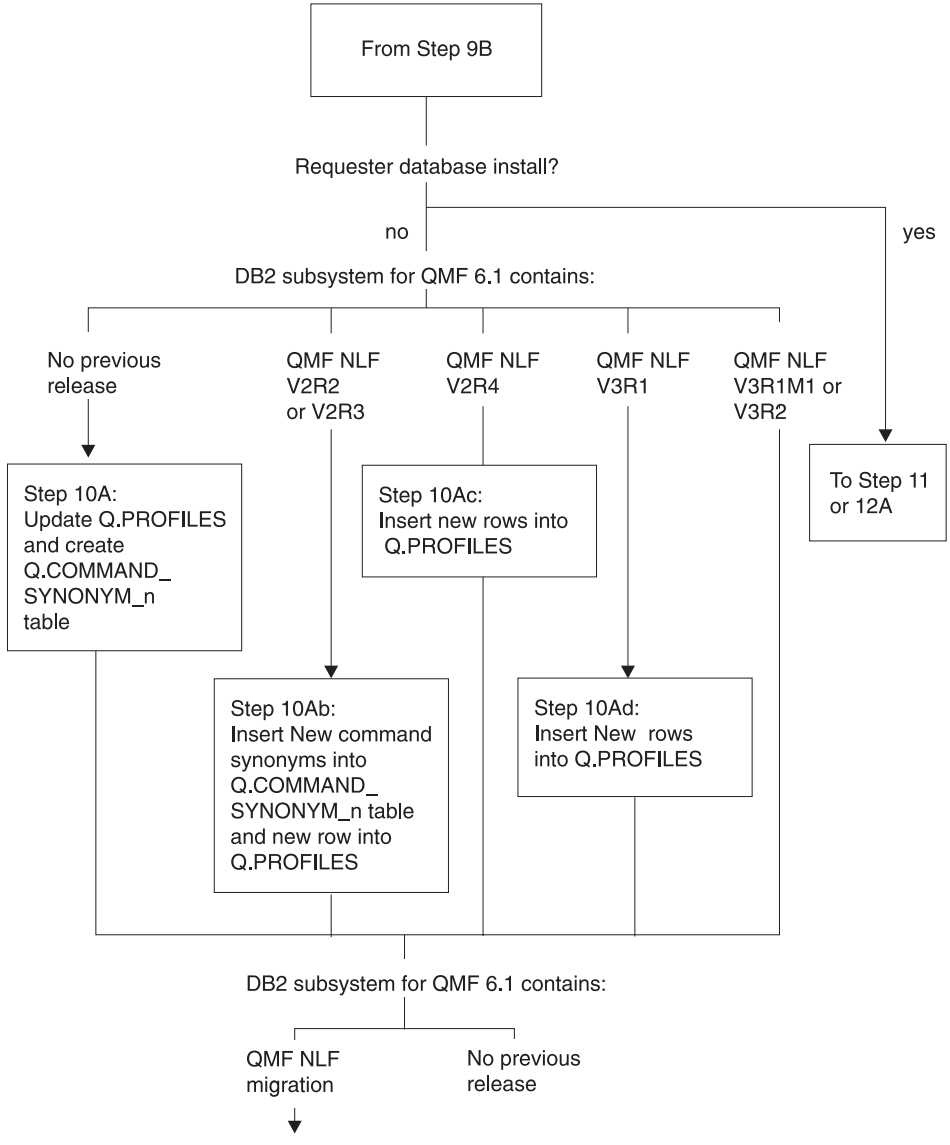


Figure 19. Installation Steps for QMF NLF -- Part 3

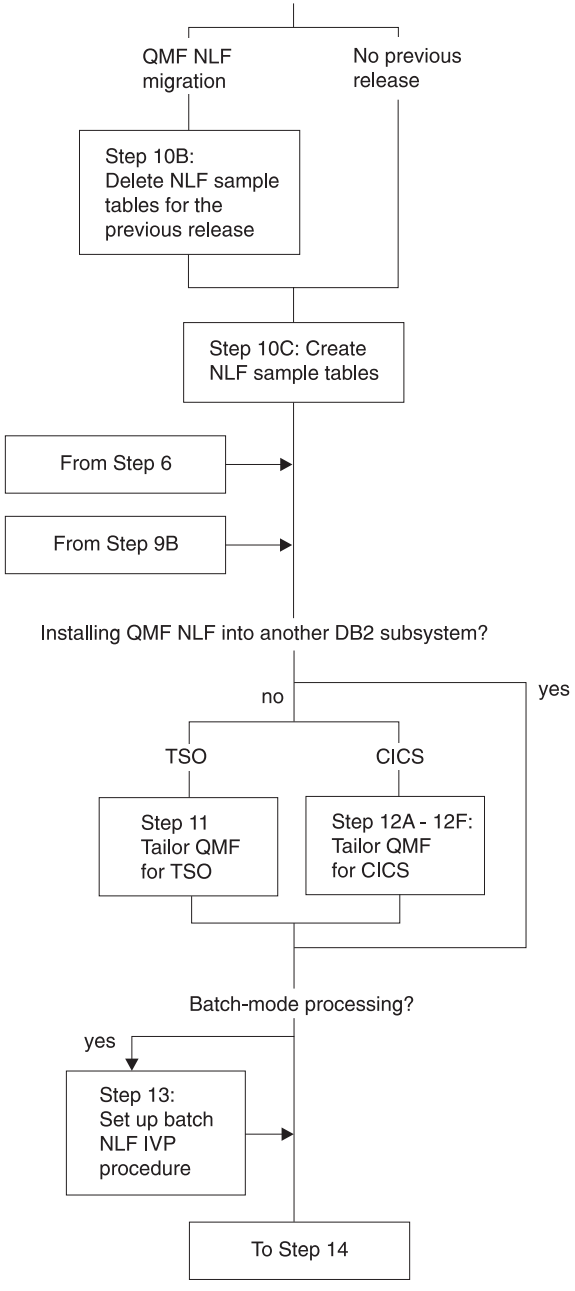


Figure 20. Installation Steps for QMF NLF -- Part 4

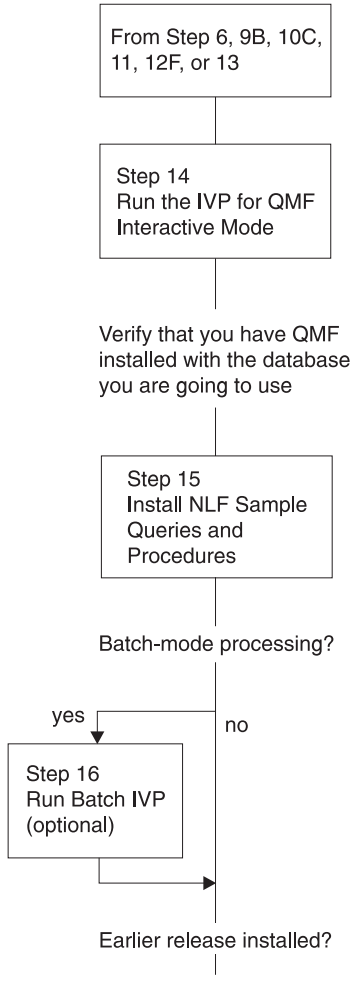


Figure 21. Installation Steps for QMF NLF -- Part 5

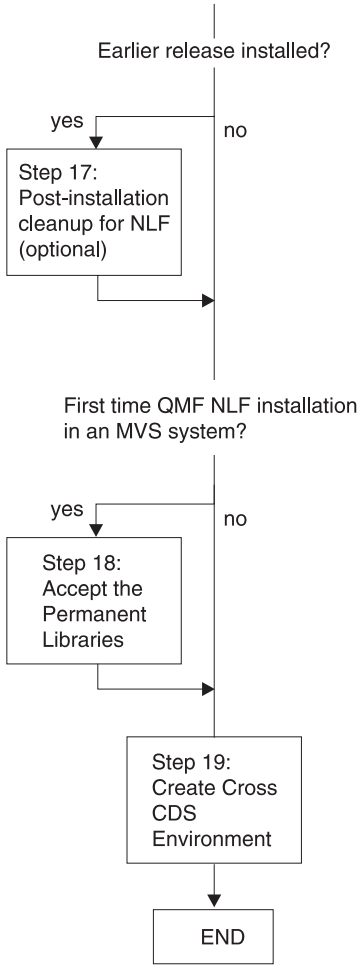


Figure 22. Installation Steps for QMF NLF -- Part 6

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. Table 19 on page 107 shows the information that you will have to provide during QMF NLF installation. You can use it as your own worksheet.

Table 19. QMF NLF Installation Parameters (Version 7 Worksheet-Part 1)

PARAMETER	VALUE
Target Library Prefix (Default = QMF720)	
Distribution Library Prefix (Default = QMF720)	
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=V3R1)	
Local DB2 Exit Library (Default=DSN710.SDSNEXIT)	
Local DB2 Load Library (Default=DSN710.SDSNLOAD)	
Communications database installed at local DB2	yes or no
Gather the following information, if the communications database is installed at the local DB2:	
Scope of Install	F (full database), S (server database), or R (requester database)
Gather the following information, if the scope of the database install is not “S”:	
Customize QMF runtime libraries	yes or no
QMF application plan ID (Default=QMF720)	
Gather the following information, if the scope of the database install is “F”, or “S”.	
QMF tablespace catalog alias (Default=QMFDSN)	
QMF tablespace catalog password (for QMF control tables)	
QMF tablespaces volume	
DB2 default punctuation	, (comma) or . (period)
Previous QMF NLF Level (Migration Installs Only)	3.1, 3.1.1, 3.2, 3.3, 6.1 or NONE

Table 20. QMF NLF Installation Parameters (Version 7 Worksheet-Part 2)

PARAMETER	PRIMARY	SECONDARY
Gather the following information, if the the scope of the database install is not “R”.		

Table 20. QMF NLF Installation Parameters (Version 7 Worksheet-Part 2) (continued)

PARAMETER	PRIMARY	SECONDARY
QMF Control Table tablespace 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)	_____	_____
Determine the following, if applicable:		
Install QMF NLF jobs in the foreground or tailor the JCL files yourself and run each job in batch?	foreground or batch	
Do you have fixed or variable-length CLIST libraries?	variable or fixed	
Do you have fixed or variable-length EXEC libraries?	variable or fixed	
Do you want SAVE DATA table space created?	yes or no	

Before performing the following steps, you must first install QMF NLF in your OS/390 environment using SMP/E as documented in the QMF NLF Program Directory.

Step 1—Provide QMF NLF installation parameters

In this step, you provide information that describes your QMF and DB2 environments. You are presented with a series of QMF Installation Dialog panels that let you "fill-in" the required data. Use the information that you provided in the worksheet, Table 19 on page 107.

Preparation

Before starting this step, consider the following requirements:

1. You must be in an active ISPF session.
2. If you have not used the QMF target library names as originally specified in DSQ1nJAL (for example, no user changes other than allowed variables, such as QMFTPRE), you must either alter the DSQ1nINS, DSQ1nIN1, and DSQ1nIN2 CLISTS, or skip to "Steps 4-8—Submit jobs manually" on page 117.

3. If you are installing QMF into another database, ensure that the QMF target libraries you are using for the installation cannot be accessed by users of other databases during installation.
4. Fixed-block SDSQCLTn and SDSQEXCn are required for Steps 1 through 3.

Execution

Enter the following:

```
TSO EXEC 'prefix.SDSQCLTn(DSQ1nINS)' 'QMFPRE(prefix)'
```

where *prefix* is the QMF target library prefix you provided in the worksheet.

Obtain the parameter input panels, in one of the following ways:

- If this is the first time that you provide installation parameters, you are presented with the first parameter input panel *automatically*.
- If this step has been completed successfully, you are presented with an initial panel offering you four options:

  P installation parameters

  T tailor install files

  I install in foreground

  X exit install dialogs

  Choose the **P** option to obtain the first parameter input panel.

As you enter the information on each panel, your input is saved in the QMF720.SDSQCLTn library under the database name that you chose.

If you leave this step before completing the last panel, your input will not be saved. The last panel asks you for job card information used to tailor the installation. If you plan to install in the foreground, you do not need to provide job card information; just enter `x` in the indicated spot on the panel.

When you have supplied the last installation parameter, the Main menu is displayed. If you want to review or modify the parameters, enter **P** and proceed through the input panels again.

If you are satisfied with your installation parameters, continue with the next step. If you prefer, you can leave the installation process at this point and return later; your installation parameters are saved. Use the information from your worksheets to work through the panels.

Main menu: The main menu is displayed if you saved any install parameters. Otherwise, it is the first panel you see when you invoke the install.

This menu lets you provide installation parameters, tailor jobs, install QMF NLF in the foreground, or exit install dialogs.

```
                          INSTALL QMF NLF -- MAIN MENU
ISPF COMMAND ===>



    CURRENTLY WORKING ON INSTALLATION INTO DB2 SUBSYSTEM DSN

    YOU CAN NOW RE-SPECIFY THE INSTALL PARAMETERS, TAILOR THE INSTALLATION
    FILES, INSTALL QMF WITH THE TAILORED FILES IN FOREGROUND, QUIT AND RUN
    THE TAILORED INSTALL FILES IN BATCH, OR QUIT AND RETURN HERE LATER.


    ENTER CHOICE HERE        ===>              ("P" - INPUT PARAMETERS,
                                                "T" - TAILOR INSTALL FILES,
                                                "I" - INSTALL IN FOREGROUND,
                                                "X" - EXIT INSTALL DIALOGS)



    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 23. Dialog main menu

Note: The last-used DB2 subsystem name is displayed on this panel.

The following options are available on this panel:

P Customizes QMF NLF install parameters, which are described in the following panels.

If you want to customize QMF NLF install parameters for another DB2 subsystem, you can ignore the DB2 subsystem name displayed in this panel and proceed to the next panel by entering “P”.

T Tailors all the required NLF install data sets for QMF 7.

These panels are not displayed in this manual.

I Installs QMF NLF in foreground. This option lets you submit the jobs (steps 4 through 8) in an online environment.

X Exits the install dialogs.

Local DB2 parameters: This panel is displayed first if you have not saved any install parameters. Otherwise, it is displayed only if you have chosen the “P” option.

```
                  INSTALL QMF NLF -- LOCAL DB2 PARAMETERS
ISPF COMMAND ===>

    LOCAL DB2 SUBSYSTEM ID    ===>

    LOCAL DB2 RELEASE LEVEL   ===>         ("31" FOR V3R1)

    LOCAL DB2 EXIT LIBRARY    ===>

    LOCAL DB2 LOAD LIBRARY    ===>


    COMMUNICATIONS DATABASE(CDB) INSTALLED AT LOCAL DB2  ===>     ("Y","N")




    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 24. Local DB2 parameters

The following options are available on this panel:

LOCAL DB2 SUBSYSTEM ID
Specify the DB2 subsystem ID in which the QMF application plan was bound (required: default is DSN).

LOCAL DB2 RELEASE LEVEL
Specify the DB2 release level of the local DB2 subsystem (required: no default).

LOCAL DB2 EXIT LIBRARY
Specify the DB2 exit library for the local DB2 subsystem (required: no default).

LOCAL DB2 LOAD LIBRARY
Specify the DB2 load library for the local DB2 subsystem (required: no default).

COMMUNICATIONS DATABASE(CDB) INSTALLED AT LOCAL DB2
Specify, if the DB2 communications database is installed at the local DB2 subsystem (required: no default).

Scope of Database Install: This panel is displayed if the DB2 release level is "23" and the communications database is installed with the local DB2.

```
                  INSTALL QMF NLF -- SCOPE OF DATABASE INSTALL
ISPF COMMAND ===>




SCOPE OF DATABASE INSTALL       ===>     ("F" - FULL DATABASE,
                                          "R" - REQUESTOR DATABASE ONLY,
                                          "S" - SERVER DATABASE ONLY)




    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 25. Database install scope

The following option is available on this panel:

SCOPE OF DATABASE INSTALL
Specify the scope of the database install. The allowable options are FULL database, REQUESTER database, and SERVER database. See "Road maps for the QMF installation process" on page 10 for more information on these options.

If you are installing QMF Version 7.2 NLF for the first time, select the FULL database install option.

QMF Parameters at Local DB2: This panel is displayed if this is not a SERVER database install.

```
                INSTALL QMF NLF -- QMF PARAMETERS AT LOCAL DB2
ISPF COMMAND ===>



    CUSTOMIZE QMF RUNTIME LIBRARIES           ===>          ("Y" OR "N")

      - INSTALL QMF PANELS
      - INSTALL QMF/GDDM MAP GROUPS
      - INSTALL QMF/GDDM SAMPLE CHARTS FORMS
      - MAKE QMF REXX EXECS AVAILABLE
      - MAKE QMF CLISTS AVAILABLE



    QMF APPLICATION PLAN ID AT LOCAL DB2      ===>




    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 26. QMF parameters at local DB2

The following options are available on this panel:

CUSTOMIZE QMF RUNTIME LIBRARIES
Specify the options, if the QMF NLF runtime libraries require customization. You only need to customize these libraries once per operating system (required: no default).

QMF APPLICATION PLAN ID AT LOCAL DB2
Specify the QMF application plan name that was bound at the local DB2 subsystem (required: no default).

DB2 and QMF Parameters: This panel is displayed next.

```
                  INSTALL QMF NLF -- DB2 AND QMF PARAMETERS
ISPF COMMAND ===>


    QMF TABLESPACES CATALOG ALIAS      ===>

    QMF TABLESPACES CATALOG PASSWORD   ===>

   QMF TABLESPACES VOLUME      ===>        ("SYSxxx" OR "AST",
                                             x is from 0 to 9,
                                             AST stands for *)

    DEFAULT PUNCTUATION                ===>             ("," OR ".")

    PREVIOUS QMF LEVEL                 ===>            ("V2R2","V2R3","V2R4",
                                                        "V3R1","V3R1M1",
                                                        "V3R2","V3R3","V6R1"
                                                        "NONE")



    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 27. DB2 and QMF parameters

The following options are available on this panel:

QMF TABLESPACES CATALOG ALIAS
Specify the VCAT name for all the QMF NLF table spaces. The VSAM data sets that associate with these QMF NLF table spaces have the high level qualifier of this alias value. If you are migrating from a previous level of QMF, use the same alias value as the previous release (required: no default).

QMF TABLESPACES CATALOG PASSWORD
Specify the password for all the QMF NLF control table spaces and index spaces that are created by the installation (optional).

QMF TABLESPACES VOLUME
Specify a volume serial number in which the QMF NLF table spaces reside (required: no default).

DEFAULT PUNCTUATION
Specify the symbol for a decimal point in DB2 for QMF NLF (required: no default).

PREVIOUS QMF LEVEL
Specify the release level of QMF NLF that you are migrating from (required: if you do not have any previous release level in the database, enter `NONE`).

Space parameters for QMF index spaces: This panel is displayed if there is no previous QMF NLF release level installed in the database.

```
             INSTALL QMF NLF -- QMF INDEXSPACES SPACE PARAMETERS
ISPF COMMAND ===>

    SPECIFY THE SIZES (IN 1K UNITS) FOR THE FOLLOWING TABLE INDEXES




       TABLE INDEX            PRIMARY        SECONDARY
       -----------            -------        ---------

       Q.COMMAND_SYNONYMX_n   ===>           ===>







    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 28. QMF index spaces space parameters

Note: The default sizes in 1 KB units are listed above.

Specify the primary and secondary allocations for the QMF index spaces. These values are used when QMF allocates all the VSAM files for these table spaces. Depending on the size of your installation, you may need to increase or decrease the default sizes to allow for additional free space.

Jobcard: This panel is always displayed as the last panel for the P option.

```
                          INSTALL QMF NLF -- JOBCARD
ISPF COMMAND ===>

    MODIFY THE JOB CARDS BELOW TO REPRESENT YOUR INSTALLATION REQUIREMENTS
    THE "USER" AND "PASSWORD" PARAMETERS MUST BE SPECIFIED IN SYSTEMS USING
    RACF. USE A USERID WITH THE APPROPRIATE AUTHORITY FOR THE DATABASE. SEE
    THE "QUERY MANAGEMENT FACILITY: INSTALLATION GUIDE FOR MVS" PUBLICATION
    FOR MORE DETAIL.


    IF YOU WILL BE PERFORMING THE INSTALLATION IN FOREGROUND AND DO NOT
    THE JCL FILES TO BE TAILORED, ENTER AN 'X' HERE.  ===>


    JOB STATEMENT INFORMATION:
    ===> //QMFINSTL JOB (ACCT),NAME,
    ===> //          CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),
    ===> //          USER=Q,PASSWORD=Q
    ===> //*

    PRESS:  ENTER TO CONTINUE    PF01 FOR HELP    PF03  TO END
```

Figure 29. Jobcard

This jobcard is used to submit all the QMF install jobs at your installation.

Step 2—Tailor the jobs

When you are satisfied with your installation parameters, select the **T** option on the "**P T I**" panel. The following steps will occur:

- A message tells you that the system is tailoring the JCL for the installation path you selected in "Step 1—Provide QMF NLF installation parameters" on page 108.
- QMF720.SDSQEXCn(DSQSCMDn), the QMF callable interface REXX EXEC, is modified to update the default values for parameters `planid` and `DB2 subsystem name`, unless you are performing a server database installation.

At the conclusion of this step, the "**P T I**" panel is displayed. You can then proceed with the installation of QMF NLF.

If you plan to tailor online, do not alter the sequence of the install JCL and control files. This is because the CLIST requires the JCL and control files to be in a particular sequence; if you alter the sequence of JCL or control files, you must modify this CLIST.

Installing QMF NLF

You can install the tailored jobs in one of two ways:

- Install the jobs in the foreground ("Step 3—Install QMF NLF in the foreground" on page 117)

- Submit each job manually (starting with "Step 4—Install QMF panels") This option enables you to meet the needs of your installation environment. It lets you modify certain installation parameter values that are provided by default.

Step 3—Install QMF NLF in the foreground

If you want to install QMF NLF in the foreground, select the **I** option on the "**P T I**" panel. A dialog panel is then displayed, requesting installation options. After you enter the information, a message tells you that the installation is proceeding.

DB2 authority

When you submit the jobs in the foreground, they are installed under your current LOGON ID. If your LOGON ID has "Q" as its authorization ID, you will need, as a minimum, DB2 authority granted by the following SQL statements:

```
GRANT USE OF BUFFERPOOL BP0 TO Q
GRANT CREATESG TO Q
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 your LOGON ID has an authorization other than "Q", you need the authority granted by the following SQL statement:

```
GRANT SYSADM TO authid
```

where *authid* is the primary authorization ID. You are now done with Step 3. Continue the installation with one of the following:

- "Step 8—Tailor NLF/QMF for TSO" on page 127
- "Step 9—Tailor NLF/QMF for CICS" on page 128

If you are installing QMF NLF into another DB2 subsystem:

- Using full or server database, go to "Step 7A—Update QMF control tables" on page 120.
- Using requester database, go to "Step 8—Tailor NLF/QMF for TSO" on page 127 or "Step 9A—Add NLF/QMF transaction ID to DB2 RCT" on page 128.

Steps 4-8—Submit jobs manually

The following steps describe how to install QMF in a batch environment.

Step 4—Install QMF panels

This step copies the expanded version of QMF panels to the panel file, QMF720.DSQPNLn.

Preparation

The job for this step is in QMF720.SDSQSAPn(DSQ1nPNL). If the tailoring performed in "Step 3—Install QMF NLF in the foreground" on page 117 was not sufficient, change the JOB statement to conform to your installation. You may also have to change the values of two parameters in the job's instream procedure:

Parameter name
description of value (default in parenthesis)

QMFTPRE
The prefix for the QMF target libraries (**QMF720**)
RGN The job step region size (**2048K**)

Execution

Run job DSQ1nPNL in library QMF720.SDSQSAPn.

Rerunning the Job

Before rerunning the job, do the following tasks:

- Delete any members that were added to the target library.
- Recover the used space by compressing the library.
- Fix the errors that caused the failure.

Step 5—Install NLF/GDDM map groups

QMF uses the GDDM screen mapping functions. This step expands the NLF/GDDM map group files located in the sample library (default name is QMF720.SDSQSAPn) to the target map group library (default name is QMF720.DSQMAPn).

Preparation

The job for this step is QMF720.SDSQSAPn(DSQ1nMAP). If the tailoring performed in "Step 3—Install QMF NLF in the foreground" on page 117 was not sufficient, change the job statement to conform to your installation requirements. You may also have to change the values of two parameters in the job's instream procedure:

Parameter name
Description of value (Default in parenthesis)

QMFTPRE
The prefix for the QMF target libraries (**QMF720**)
RGN The job step region size (**2048K**)

Execution

Run job DSQ1nMAP (in the library QMF720.SDSQSAPn).

Rerunning the job

Before rerunning the job, do the following:

- Delete any members that were added to the target library.
- Recover the used space by compressing the library.
- Fix the errors that caused the failure.

Step 6—Converting REXX EXEC or CLIST records

This step converts QMF REXX EXEC or QMF CLIST records from fixed to variable length.

Step 6A—Converting QMF REXX EXEC records: fixed length to variable

Note: You must have TSO/E Version 2 (or later) installed to use REXX EXECs in an OS/390 environment.

Later in this procedure, you must allocate the QMF exec library (QMF720.SDSQEXCn) as a SYSEXEC data set. The library should be concatenated to other exec libraries.

Example

In the following JCL the library QMF720.SDSQEXCn is concatenated to an EXEC library named SYS2.EXEC.

```
//SYSEXEC  DD DSN=SYS2.EXEC,DISP=SHR
//         DD DSN=QMF720.SDSQEXCn,DISP=SHR
```

The QMF exec library contains fixed-length records. It can only be concatenated to other exec libraries that have fixed-length records. If they have variable-length records, you must create a copy of the QMF library with variable-length records.

For more information, see *Developing QMF Applications*.

Preparation: The job for this step is QMF720.SDSQSAPn(DSQ1nJVE). Change 'xxxxxx' in the DSQTEVB.SYSUT2 statement to the serial number of the volume for the copy of the library. If the tailoring performed in "Step 2—Tailor the jobs" on page 116 was not sufficient, you may want to do the following tasks:

- Change the job statement to conform to your installation.
- Change, if necessary, the value of the QMFTPRE parameter in the job's instream procedure. This value is the prefix for the QMF libraries (default is QMF720).
- Leave the exec procedure parameter blank; it is used by the job when the procedure is called.
- Change, if necessary, the name of this library.

Execution: Run job DSQ1nJVE.

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

Step 6B—Converting QMF CLISTs records: fixed length to variable

Later in this procedure, you must allocate the QMF CLIST library (QMF720.SDSQCLTn) as a SYSPROC data set. The library should be concatenated to other CLIST libraries.

Example

In the following JCL the library QMF720.SDSQCLTn is concatenated to a CLIST library named SYS2.CLIST.

```
//SYSPROC  DD  DSN=SYS2.CLIST,DISP=SHR
//         DD  DSN=QMF720.SDSQCLTn,DISP=SHR
```

The QMF CLIST library contains fixed-length records. It can only be concatenated to the other CLIST libraries that have fixed-length records. If they have variable-length records, you must create a copy of the QMF library with variable-length records.

Preparation: The job for this step is QMF720.SDSQSAPn(DSQ1nJVC). Change 'xxxxxx' in the DSQTIVB.SYSUT2 statement to the serial number of the volume for the copy of the library. If the tailoring performed in "Step 2—Tailor the jobs" on page 116 was not sufficient, you may want to do the following steps:

- Change the job statement to conform to your installation.
- Change, if necessary, the value of the QMFTPRE parameter in the job's instream procedure. This value is the prefix for the QMF libraries (default is QMF720).
- Leave the CLIST procedure parameter blank. It is used by the job when the procedure is called.
- Change, if necessary, the name of this library.

Execution: Run job DSQ1nJVC.

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

If you are doing a requester database install, go to "Step 8—Tailor NLF/QMF for TSO" on page 127 or "Step 9A—Add NLF/QMF transaction ID to DB2 RCT" on page 128.

Step 7A—Update QMF control tables

The substep (7Aa, 7Ab, 7Ac, 7Ad) you perform depends on the type of migration installation you are running:

- If no previous QMF NLF release is installed, do "Substep 7Aa—without a previous QMF NLF release".
- If you are running a QMF NLF 2.2 or 2.3 migration, do "Substep 7Ab—with QMF NLF 2.2 or 2.3" on page 122.
- If you are running a QMF NLF 2.4, do "Substep 7Ac—with QMF NLF 2.4" on page 123.
- If you are running a QMF NLF 3.1, do "Substep 7Ad—with QMF NLF 3.1" on page 124.
- If QMF NLF 3.1.1, QMF NLF 3.2, QMF NLF 3.3, or QMF NLF 6.1 is in this DB2 subsystem, skip to "Step 7B—Delete earlier QMF NLF sample tables" on page 125.

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 7Aa—without a previous QMF NLF release

Perform this step if you do not have a previous QMF NLF installed.

On this step, you do the following:

- Add NLF entries in the Q.PROFILES table. The job is QMF720.SDSQSAPn(DSQ1nUPO).
- Create, for the NLF environment, a command synonyms table named Q.COMMAND_SYNONYM_n. The job is QMF720.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 (**QMF720**)
DB2EXIT
Name of the DB2 exit library (**DSN710.SDSNEXIT**)
DB2LOAD
Name of the DB2 program library (**DSN710.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 then examine SYSTSPRT and SYSPRINT for error messages.

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

Substep 7Ab—with QMF NLF 2.2 or 2.3

Perform this step only if you are migrating from QMF NLF 2.2 or 2.3.

This job adds your NLF equivalents of the IRM and LAYOUT command synonyms to the Q.COMMAND_SYNONYM_n table, and update the Q.PROFILES control table.

Preparation: The job used in this step is QMF720.SDSQSAPn(DSQ1nICS). Change the job statement to conform to your installation. Also, make the necessary changes to the following parameters in the job's instream procedure:

Parameter name
Description of value (Default in parenthesis)

QMFTPRE
The prefix name of the QMF target libraries (**QMF720**)
DB2EXIT
Name of the DB2 exit library (**DSN710.SDSNEXIT**)
DB2LOAD
Name of the DB2 program library (**DSN710.SDSNLOAD**)
RGN Job-step region size (**2048K**)

DB2 authority: If you are the user Q , you have the necessary authority to run the job.

If you are not the user Q, run the following query, to get the necessary authority:

```
GRANT INSERT ON TABLE Q.COMMAND_SYNONYM_n TO authid
```

where *authid* is your primary authorization ID.

Execution: Run job QMF720.SDSQSAPn(DSQ1nICS).

Review SYSTERM for completion messages. If errors occur then examine SYSTSPRT and SYSPRINT for error messages.

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

Substep 7Ac—with QMF NLF 2.4

Perform this step only if you are migrating from QMF NLF V2R4.

This job updates the Q.PROFILES control table.

Preparation: The job used in this step is QMF720.SDSQSAPn(DSQ1nUP1). Change the job statement to conform to your installation. Change, if necessary, 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 (**QMF720**)
DB2EXIT
Name of the DB2 exit library (**DSN710.SDSNEXIT**)
DB2LOAD
Name of the DB2 program library (**DSN710.SDSNLOAD**)
RGN Job-step region size (**2048 KB**)

DB2 authority: If you are the user Q, you have the necessary authority to run the job.

If you are not the user Q, run the following query to get the necessary authority:

```
GRANT INSERT ON TABLE Q.PROFILES TO authid
```

where *authid* is your primary authorization ID.

Execution: Run job QMF720.SDSQSAPn(DSQ1nUP1).

Review SYSTERM for completion messages. If errors occur then examine SYSTSPRT and SYSPRINT for error messages.

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

Substep 7Ad—with QMF NLF 3.1

Perform this step only if you are migrating from QMF NLF V3R1.

This job updates the Q.PROFILES control table.

Preparation: The job used in this step is QMF720.SDSQSAPn(DSQ1nUP2). Change the job statement to conform to your installation's requirements. Change, if necessary, 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 (**QMF720**)
DB2EXIT
Name of the DB2 exit library (**DSN710.SDSNEXIT**)
DB2LOAD
Name of the DB2 program library (**DSN710.SDSNLOAD**)
RGN Job-step region size (**2048K**)

DB2 authority: If you are the user Q, you have the necessary authority to run the job.

If you are not the user Q, run the following query, to get the necessary authority:

```
GRANT INSERT ON TABLE Q.PROFILES TO authid
```

where *authid* is your primary authorization ID.

Execution: Run job QMF720.SDSQSAPn(DSQ1nUP2).

Review SYSTERM for completion messages. If errors occur then examine SYSTSPRT and SYSPRINT for error messages.

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

Step 7B and 7C—Establish the QMF NLF sample tables

Skip both steps 7B and 7C if either of the following apply:

- The NLF is the upper case feature (UCF).
- You already have the sample tables installed from an earlier Version 2 (any 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 run by the step ends.

Step 7B—Delete earlier QMF NLF sample tables

Do this step if you are installing QMF 7 NLF into a DB2 subsystem that also contains a previous release of QMF NLF. Otherwise, skip to "Step 7C—Create the NLF sample tables" on page 126.

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 7 NLF.

Preparation

The job used in this step is QMF720.SDSQSAPn(DSQ1nDSJ). If the tailoring performed in "Step 3—Install QMF NLF in the foreground" on page 117 was not sufficient, change the job statement to conform to your installation's requirements. Change, if necessary, 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 (**QMF720**)

DB2EXIT
: Name of the DB2 exit library (**DSN710.SDSNEXIT**)

DB2LOAD
: Name of the DB2 program library (**DSN710.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 QMF720.SDSQSAPn). Review SYSTERM for completion messages. If errors occur then examine SYSTSPRT and SYSPRINT for error messages.

Rerunning the job

If the job fails, you can correct the error and rerun it. It may, however, fail because the tables it attempts to drop have already been dropped.

Step 7C—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 into which you are installing the QMF NLF.

Preparation

The job for this step is QMF720.SDSQSAPn(DSQ1nIVS). If the tailoring performed in step 3 was not sufficient, change the job statement to fit your installation requirements. Change, if necessary, 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 (**QMF720**)

DB2EXIT
Name of the DB2 exit library (**DSN710.SDSNEXIT**)

DB2LOAD
Name of the DB2 program library (**DSN710.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 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 OS390 Installation Guide*.

DB2 authority

If you are the user Q, you need, as a minimum, 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 QMF720.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 the job.

If you are installing QMF NLF into another database, go to "Step 12—Set Up NLF batch job to run batch IVP (optional)" on page 135.

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 9—Tailor NLF/QMF for CICS" on page 128.

Step 8—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 that outlined in Chapter 4, "Tailoring QMF for TSO", on page 47.

- 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=QMF720.DSQMAPn,DISP=SHR
    ```
  - The statement to concatenate to the ISPPLIB DD statement is:
    ```
    //ISPPLIB  DD DSN=QMF720.SDSQPLBn,DISP=SHR
    ```
  - The statement to concatenate to the ISPMLIB DD statement is:
    ```
    //ISPMLIB  DD DSN=QMF720.SDSQMLBn,DISP=SHR
    ```
  - The statement to concatenate to the SYSPROC DD statement is:
    ```
    //SYSPROC  DD DSN=QMF720.SDSQCLTn,DISP=SHR
    ```
  - The statement to concatenate to the SYSEXEC DD statement is:
    ```
    //SYSEXEC  DD DSN=QMF720.SDSQEXCn,DISP=SHR
    ```
  - The statement to concatenate to the DSQPNLn DD statement is:
    ```
    //DSQPNLn  DD DSN=QMF720.DSQPNLn,DISP=SHR
    ```
- The statement to start QMF with ISPF looks like the following:
  ```
  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 30. 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 9—Tailor NLF/QMF for CICS

You can run this step after the QMF product has been tailored for CICS as described in Chapter 6, "Tailoring QMF for CICS", on page 69. If you are migrating from QMF 3.1, you need to run all the steps except "Step 9Da—update CICS control tables (CICS V2 only)" on page 130. (For information on CICS migration considerations, see *Installing and Managing QMF for OS/390*.)

Step 9A—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 QMF720.SDSQSAPn(DSQ1nRCT).

After the RCT is updated with information describing the QMF transaction to DB2, you must then regenerate your RCT.

Step 9B—Link-edit with DFHEAI and DFHEAI0

QMF uses the CICS command-level application programming interface to operate under CICS. You must link-edit QMF with the exec interface modules DFHEAI and DFHEAI0 before you can run any QMF programs. To include CICS interface modules DFHEAI and DFHEAI0, you must run this step each time you apply QMF service.

Substep 9Ba— Link-edit QMF with CICS command interface modules

This job link-edits QMF NLF modules with CICS command level support modules DFHEAI and DFHEAI0.

Preparation: The job used in this step is QMF720.SDSQSAPn(DSQ1nLNK). Change the job statement to conform to your installation. Change, if necessary, the values for the installation parameters in the job's instream procedure:

Table 21. Installation parameters for DSQ1nLNK

Parameter name	Description of value	Default
QMFTPRE	The prefix name of the QMF target libraries	QMF720
REG	The job-step region size	4096
OUTC	The job output class	*
CLOAD	The name of the CICS load library	CICS.LOADLIB

After completing this job, examine the listing and ensure that all modules have been link-edited successfully.

Note: You must rerun this job if any of the modules are changed by PTF.

Substep 9Bb—translate, assemble and link-edit the QMF supplied governor

Preparation: The job used in this step is QMF720.SDSQSAPn(DSQ1nGLK). Change the job statement to conform to your installation. Change, if necessary, the values for the installation parameters in the job's instream procedure:

Table 22. Installation parmaters for DSQ1nGLK

Parameter name	Description of value	Default
QMFTPRE	The prefix name of the QMF target libraries	QMF720

Table 22. Installation parmaters for DSQ1nGLK (continued)

Parameter name	Description of value	Default
REG	The job-step region size	4096
OUTC	The job output class	*
CLOAD	The name of the CICS load library	CICS.LOADLIB
CMACS	The name of the CICS macro library	CICS.MACLIB
SUFFIX	The CICS ASM Translator suffix	1$
ASMBLR	The name of the Assembler	IEV90
WPRL	The name of the work volume unit	SYSDA

Step 9C—load QMF/GDDM map sets to the ADMF data set

Preparation: The job used in this step is QMF720.SDSQSAPn(DSQ1nADM). Change the job statement to conform to your installation. Change, if necessary, the values for the installation parameters in the job's instream procedure.

Table 23. Installation parameters for DSQ1nADM

Parameter name	Description of value	Default
QMFTPRE	The prefix name of the QMF target libraries	QMF720
REG	The job-step region size	2048
GDDM	The name of the GDDM ADMF data set	GDDM.ADMF

Step 9Da—update CICS control tables (CICS V2 only)

Before you can run the NLF/QMF feature under CICS, QMF entries must be defined as follows:

FCT (file control table): Describes the QMF panel file that contains NLF/QMF help and screen definitions. Add or copy member DSQ1nFCT (in library QMF720.SDSQSAPn) into existing FCT entries on your CICS system.

PCT (program control table): Describes the QMF transaction name for this NLF/QMF. Add or copy member DSQ1nPCT (in library QMF720.SDSQSAPn) into existing PCT entries on your CICS system.

PPT (processing program table): Describes the QMF programs that contain NLF/QMF constants and messages. Add or copy member DSQ1nPPT (in library QMF720.SDSQSAPn) into existing PPT entries on your CICS system.

After you include or copy members into your CICS system, assemble and link-edit.

Step 9Db—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 QMF720.SDSQSAPn(DSQ1nCSD). Change the job statement to conform to your installation. Change, if necessary, the values for the installation parameters in the job's instream procedure:

Table 24. Installation parameters for DSQ1nCSD

Parameter name	Description of value	Default
QMFTPRE	The prefix name of the QMF target libraries	QMF720
REG	The job-step region size	2048
OUTC	The job output class	*
CLOAD	THe name of the CICS load library	CICS.LOADLIB
CCSD	The name of the CICS CSD data set	CICS.DFHCSD

Step 9E—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=QMF720.DSQPNLn,DISP=SHR
```

where *n* is the NLF character.

Step 9F—Run the IVP

Run the IVP as indicated in "Step 33 (for CICS)—Run the IVP" on page 87, changing the following names:

- QMF320.DSQSAMPE to QMF720.SDSQSAPn
- DSQ1EIVC to DSQ1nIVC

where *n* is the NLF character.

Step 10—Tailoring QMF NLF for a Workstation Database Server (optional)

QMF support for Workstation Database Server is optional. You need to 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 OS/390, you need to verify that you have followed the steps needed 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 7, "Tailoring QMF for Workstation Database Servers", on page 75.

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 10A—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 QMF720.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='QMF720',          Prefix for QMF target libraries
//             DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for OS/390 subsystem ID.
4. Submit job QMF720.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 10B—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 QMF720.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='QMF720',          Prefix for QMF target libraries
//             DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for OS/390 subsystem ID.
4. Submit job QMF720.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 QMF720.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='QMF720',          Prefix for QMF target libraries
//             DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for OS/390 subsystem ID.
4. Submit job QMF720.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 tablespace from a Workstation Database Server.

1. Edit QMF720.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='QMF720',          Prefix for QMF target libraries
//             DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
//             DB2LOAD='DSN710.SDSNLOAD'  DB2 program library name
```

3. Change *DSN* in `SYSTEM(DSN)` to your DB2 UDB for OS/390 subsystem ID.
4. Submit job QMF720.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 11—Tailoring QMF NLF for a DB2 UDB for iSeries server (optional)

QMF support for DB2 UDB for iSeries Database Servers is optional. You need to perform the steps described in this step 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 OS/390, you need to 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 QMF720.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='QMF720', Prefix for QMF target libraries
// DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
// DB2LOAD='DSN710.SDSNLOAD' DB2 program library name
```

3. Change in SYSTEM() to your DB2 for OS/390 subsystem ID.
4. Carefully read the comments in the job and make any necessary changes.
5. Submit job QMF720.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 QMF720.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='QMF720', Prefix for QMF target libraries
// DB2EXIT='DSN710.SDSNEXIT', Exit DB2 library name
// DB2LOAD='DSN710.SDSNLOAD' DB2 program library name
```

3. Change in SYSTEM() to your DB2 for OS/390 subsystem ID.

4. Carefully read the comments in the job and make any necessary changes.
5. Submit job QMF720.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 12—Set Up NLF batch job to run batch IVP (optional)

For the NLF, you must modify the TSO logon procedure described in "Step 18—Set up QMF batch job to run batch IVP (optional)" on page 54. Modify the ISPSTART command at the end of that procedure as follows:

```
ISPSTART PGM(DSQQMFn) NEWAPPL(DSQn) PARM(DSQSMODE=B,DSQSRUN=Q.DSQ1nBAT)
```

Step 13—Running the IVP for QMF interactive mode

See "Step 33 (for TSO)—run the IVP" on page 85 and "Step 33 (for CICS)—Run the IVP" on page 87 for information on running the IVP. The NLF IVP (DSQ1nIVP) (found in the library QMF720.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 QMF720. If this is not your prefix, change QMF720 to match your prefix wherever it appears in the DSQ1nIVP procedure.

```
IMPORT PROC FROM 'QMF720.SDSQSAPn(DSQ1nIVP)'
RUN PROC
```

Step 14—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 14A—Deleting the existing sample queries and procedures"
- "Step 14B—Installing the national language sample queries and procedures" on page 136

Step 14A—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 7.2 sample library, QMF720.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 QMF720. If this is not your prefix, change QMF720 to match your prefix wherever it appears in the DSQ1nSQD procedure.

```
IMPORT PROC FROM 'QMF720.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, you can proceed to the next step.

Step 14B—Installing the national language sample queries and procedures

To install the National Language sample queries and procedures, import and run the QMF procedure in QMF720.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 QMF720. If this is not your prefix, change QMF720 to match your prefix wherever it appears in the DSQ1nSQI procedure.

```
IMPORT PROC FROM 'QMF720.SDSQSAPn(DSQ1nSQI)'
RUN PROC
```

If you are not the user Q, see Step 34—Install the QMF application queries and application objects (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 15—Running the batch-mode IVP (optional)

See "Step 35—Run the batch-mode IVP (optional)" on page 91 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 7 batch IVP.

Step 16—Post-installation cleanup

See "Step 36—Clean up after install" on page 92 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 7.2 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 31. Libraries to be deleted from earlier QMF NLF releases

Step 17—Accept the permanent libraries

Perform this step if this is a first-time QMF NLF install for language *n* in an OS/390 system.

The job name for this step is DSQ1nJAC, which invokes procedure DSQ1nJSM or the SMP/E procedure used at your installation. See “Step 37—Accept the permanent libraries” on page 96 for information on running the SMP/E ACCEPT step.

Step 18—Create a cross-CDS environment

Skip this step if no maintenance changes were made to modules common to base QMF Version 7.2 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 QMF720.SDSQSAPn). The input to this job is located in member DSQ1nJCL

(in library QMF720.SDSQSAPn).

Chapter 11. Binding QMF Version 7.2 Packages at a Remote Server

In order for a QMF Version 7.2 requester installation to be able to communicate to a server, QMF Version 7.2 packages must be present at the server. If a complete QMF Version 7.2 new or migration installation was performed at the server, communications can be started and nothing further needs to be done. But, for those servers containing QMF Version 3.3 or above where migration is not a current option, you can run the job DSQ1BPKG found in the QMF720.SDSQSAPE dataset. This job binds QMF Version 7.2 packages at any server specified (provided QMF Version 3.3 or higher is detected at the server). Read, tailor and submit job DSQ1BPKG to perform the binds. Check the job output for error messages and rerun the job as necessary.

Scenario for use: Local DB2 for OS/390 subsystem, DB2G, is migrated from QMF Version 3.3 to QMF Version 7.2. QMF users in subsystem DB2G regularly communicate with a DB2 for VM server, SQLV61A, which contains QMF Version 3.3. The DB2 for VM DBA cannot perform a QMF migration to Version 7.2 at the VM server. In order for the QMF Version 7.2 installation in DB2G to communicate with QMF on SQLV61A, job DSQ1BPKG must be run to bind packages at the DB2 for VM server.

Part 2. Installing QMF on VM/ESA

Chapter 12. Introduction

The Query Management Facility (QMF) is a query and report writing program for users who have little or no data processing knowledge, as well as those with much experience in the field. This program allows users to query data and to generate online reports and charts based on the resulting data.

Overview of QMF

QMF runs under the IBM® Virtual Machine (VM), and accesses data through DB2 for VM. Provided you are not using remote unit of work with QMF Version 7.2, any data retrieved, updated, or deleted from the database is handled by DB2 for VM. QMF uses the Graphical Data Display Manager (GDDM®) to display panels, and the Interactive System Productivity Facility (ISPF) to display application panels.

If you are a Shared File System (SFS) directory user you can assume that whenever the term "minidisk" is used in this manual the same conditions apply to a "SFS directory".

QMF objects

QMF works with the following objects:

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

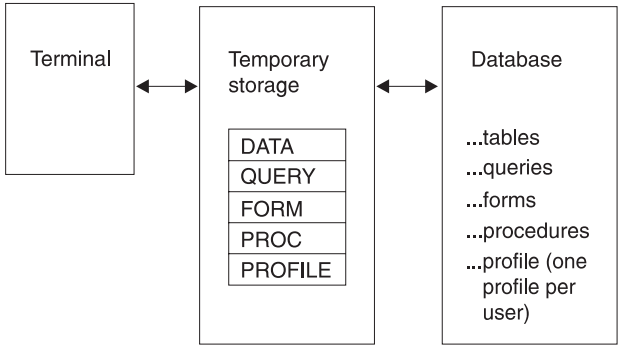
Query Specifies the data you want and the action you want to perform.

Form Describes how retrieved data should be formatted 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 an individual user's session.

These objects are brought into a temporary storage area where users can change and display reports or charts online without actually changing the database. When the user is satisfied with the changes, the objects can be saved in the database, as shown in the following diagram:



Overview of QMF with remote unit of work

With the remote unit of work function, QMF can access relational data in a remote DB2 for OS/390, DB2 for VM, DB2 for VSE, DB2 Workstation or DB2 iSeries database server. Once connected to a location you can access the data and QMF objects at that location in much the same way you would access data and objects without a remote unit of work connection.

If you use the start-up program parameter DSQSDBNM or the QMF CONNECT command to specify a remote location to connect to, all subsequent QMF commands that access the database are directed to that location.

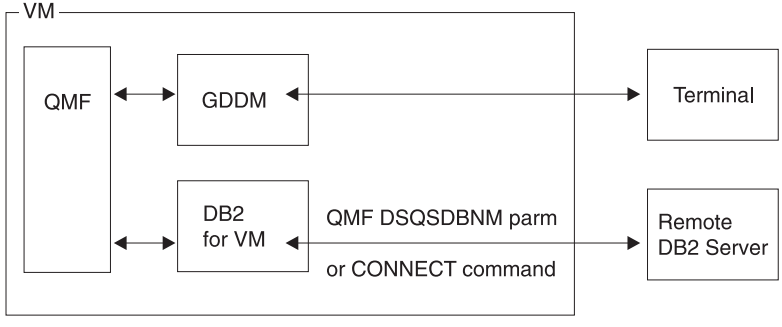


Figure 32. QMF Relationship to VM, DB2 for VM, and GDDM

Note: Before you can connect to a location you must have QMF installed in the database at that location.

Terminology

You are installing QMF Version 7 Release 2 (for brevity referred to as QMF Version 7.2). Where QMF appears without a qualifier, (For example, "QMF will run on ... ") we mean QMF Version 7.2.

Overview of the installation process

QMF installation involves three object groups

- QMF load modules
- QMF control tables, catalog views, and sample tables
- QMF database packages

Where the objects reside

The load modules are saved into a discontiguous shared segment (DCSS) that can be used from the VM user machines where users invoke QMF. The control tables, catalog views, sample tables, and packages are installed in each database that you want to access.

Local and remote installation

In a local installation you install QMF database objects into a DB2 for VM database in the same system into which you are installing QMF.

In a remote installation you install QMF database objects into a DB2 database in another system. The application requester and server are not required to reside in the same system, but a system can be configured as both.

Connecting to a remote database from VM

If you plan to connect to a remote DB2 database from VM (with the DSQSDBNM startup parameter or the CONNECT command) perform the following task:

- Install the QMF control tables, catalog views, sample tables, and packages in the DB2 database you want to connect to. For a DB2 VM, DB2 workstation, or DB2 iSeries database installation, this can be accomplished using the installation jobs that are outlined in this chapter. For installation into a DB2 OS/390 or DB2 VSE database, refer to the appropriate operating system installation instructions in this book.

  Note: If you do not have QMF installed in your local DB2 for VM database you must use the DSQSDBNM startup parameter to connect to the DB2 database during the QMF session initialization.
- If QMF Version 3.2 or higher is already installed in the remote database you want to connect to, you can simply install the new required QMF Version 7.2 database packages into that remote database. Use the job outlined in "Step 10—Load QMF database packages to a remote server (optional): DSQ2BPKB" on page 182.

Chapter 13. Planning for Installation

This chapter describes the hardware, program products, and storage required to install and run QMF. It presents an installation planning overview.

Hardware requirements

QMF runs on any processor supported by the VM operating system and DB2 for VM. QMF can access all direct-access storage devices (DASD) supported by VM and DB2 for VM, and all terminals supported by GDDM. For information about terminals supported by the GDDM, consult the GDDM general information manual.

In order to use the Double Byte Character Set (DBCS) you must have the IBM 5550 Kanji workstation, or equivalent.

Prerequisite software

The following table lists the program products with the minimum release levels required to support QMF for VM Version 7.2. Later releases that are not available at the QMF Version 7.2 announcement time are not supported unless specifically stated otherwise.

Table 25. Prerequisite Software For QMF For VM/ESA Version 7.2

Required product	Version and release	Number
IBM VM/ESA	Version 2.2	5654–030
SQL/DS for VM	Version 3.5	5688–103
GDDM/VMXA or	Version 2.3	5684–007
GDDM/VM	Version 3.1.1	5684–168

The following table lists the program products with the minimum release levels required to support optional functions for QMF for VM Version 7.2. Later releases that are not available at the QMF Version 7.2 announcement time are not supported unless specifically stated otherwise.

Table 26. Prerequisite software for optional functions for QMF for VM V7.2

Product	Version and release	Number
ISPF	Version 3.2	5684–043
CHARTS (Interactive Chart Utility):		
GDDM — PGF (for GDDM/VMXA Version 2.3) or	Version 2.1.1	5668–812
GDDM — PGF (for GDDM/VM Version 3.1.1)	Version 2.1.2	5668–812
Default editor for QMF EDIT command, display printed report application (DPRE), ISPF command, and DXT/End User Dialogs bridge support:		
ISPF/Program Development Facility for VM	Version 3.2	5684–123
QMF Document Interface:		
VM/SP System Product Editor (XEDIT)		
IBM OfficeVision/VM	Version 1.2	5684–084
ISPF/Program Development Facility for VM	Version 3.2	5684–123
Callable Interface Programs using the callable interface can be written in:		
IBM C/370 Compiler and	Version 2	5688–187
C/370 Library	Version 2	5688–188
IBM HLASM	Version 1.1 or Version 1.2	5696–234
VS COBOL II Compiler and Library	Version 1.4	5688–023
VS COBOL II Compiler, Library and Debugging Facility	Version 1.4	5668–958
AD/Cycle COBOL/370	Version 1.1	5688–197
IBM COBOL for MVS and VM	Version 1.2	5688–197
AD/Cycle C/370 Compiler	Version 1.1	5688–216
VS FORTRAN (REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)	Version 2.5	5688–806
OS PL/I	Version 2.2.3	5668–909

Table 26. Prerequisite software for optional functions for QMF for VM V7.2 (continued)

Product	Version and release	Number
IBM PL/I for MVS and VM	Version 1.1.1	5688–235
REXX: TSO Extensions (TSO/E)(REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)	Version 2.1	5685–025
REXX(REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)	In VM/ESA	
Assembler H	Version 2.1	5668–962
IBM C/C++ for MVS/ESA (In conjunction with Language Environment for MVS and VM (MVS feature)).	Version 3	5655–121
User Edit Routines can be written in:		
IBM HLASM	Version 1	5696–234
VS COBOL II Compiler and Library	Version 1.4	5688–023
COBOL/370 Compiler and Library	Version 1.1	5688–197
VS COBOL II Compiler and Library	Version 1.3.1	5688–023
VS COBOL II Compiler, Library and Debugging Facility	Version 1.3.1	5668–958
IBM COBOL for MVS and VM	Version 1. 2	5688–197
OS PL/I	Version 2.3	5668–909
IBM PL/I for MVS and VM	Version 1.1.1	5688–235
Assembler H or standard assembler	Version 2.1	5668–962
Governor Exit Routine		
IBM HLASM	Version 1	5696–234
QMF for Windows:		
Microsoft Windows XP		
Microsoft Windows ME		
Microsoft Windows 2000		

Table 26. Prerequisite software for optional functions for QMF for VM V7.2 (continued)

Product	Version and release	Number
Microsoft Windows 95 or 98		
Microsoft Windows NT	Version 4.0	
IBM APPC Networking Services for Windows, or	Version 1	
Microsoft SNA Server, or	Version 2, Version 2.1, or Version 2.11	
Novell Netware for SAA, or	Version 2	
Attachmate EXTRA! APPC Client	Version 3.11	
Remote Unit of Work (VM)		
Connection to remote DB2 for VM on VM DRDA Application Server:		
At the local DB2 for VM location:		
SQL/DS for VM	Version 3.5	5688–103
QMF for VM	Version 7.2	5697-F42
At the remote DB2 for VM database:		
SQL/DS for VM	Version 3.5	5688–103
QMF for VM	Version 7.2	5697-F42
Connection to remote DB2 for MVS/ESA DRDA Application Server:		
At the local DB2 for VM database:		
SQL/DS for VM	Version 3.5	5688–103
QMF for VM	Version 7.2	5697-F42
At the remote DB2 for MVS/ESA location:		
DB2 for MVS	Version 3.1	5685–DB2
QMF for OS/390	Version 7.2	5675-DB2
Connection to remote DB2 for VSE DRDA Application Server:		
At the local DB2 for VM location:		
SQL/DS for VM	Version 3.5	5688–103
QMF for VM	Version 7.2	5697-F42
At the remote DB2 for VSE/ESA location:		
SQL/DS for VSE	Version 3.5	5688–103

Table 26. Prerequisite software for optional functions for QMF for VM V7.2 (continued)

Product	Version and release	Number
QMF for VSE	Version 7.2	5697-F42
Connection to DB2 PE, DataJoiner, Common Server, iSeries:		
At the local DB2 for VM location:		
SQL/DS for VM	Version 3.5	5697-F42
QMF for VM	Version 7.2	5697-F42
At the remote database configured for APPC communications:		
DB2 Parallel Edition for AIX or	Version 1. 2	5765–328
DataJoiner for AIX or	Version 1.2	84H1212
DB2 for Windows NT or	Version 2.1	53H7474
DB2 for AIX or	Version 2.1	41H2128
DB2 for HP-UX or	Version 2.1	10H2366
DB2 for Solaris or	Version 2.1	10H2421
DB2 for SCO OpenServer or	Version 2.1	79H5359
DB2 for SINIX or	Version 2.1	79H4133
DB2 for iSeries	Version 4.4	5769–ST1

Virtual storage requirements

All QMF modules (31-bit shared segment) use approximately 2.8 MB total. User storage required to run QMF requires approximately 0.5 to 1 MB. You can allocate storage for both purposes above 16 MB. Additional storage is required for other applications. For example, if you run in a standard CMS environment with ISPF and GDDM, you need approximately 6 MB.

If users generate complex reports or use CMS execs to run other functions within a QMF session more storage may be required. Graphics (for example, the CHART function) requires additional storage.

Refer to the Program Directory on the ISD tape for information on Discontiguous shared segments (DCSS) storage requirements or on disk storage requirements.

Required DB2 for VM knowledge

Although QMF has been designed to be installed with a minimum of DB2 for VM knowledge, some knowledge of DB2 for VM is required.

General:

- Identifying programs and userids through the CONNECT command. Understand how the CONNECT command can be used to acquire DBA authority. For more details, see *DB2 Server for VSE & VM Database Administration*
- What a DBSPACE is and the meaning of a PUBLIC or PRIVATE DBSPACE. DBSPACEs are discussed briefly in "QMF DBSPACE requirements" on page 156. For more details, see *DB2 Server for VSE & VM Database Administration*
- CREATE, INSERT, and GRANT SQL statements. These SQL statements are used in the QMF installation procedure. Information on what these statements do and how to change them is found in *DB2 Server for VSE & VM SQL Reference*
- Preprocessing a program. All application programs that contain SQL commands must be preprocessed. Information about preprocessing a program is in *DB2 Server for VSE & VM Application Programming*
- Familiarity with the terms remote unit of work, application requester, and application server.

  remote unit of work
  QMF supports remote unit of work. With remote unit of work you can connect to locations that have QMF installed in either the DB2 or the DB2 for VM database system.

  application requester and server
  If you use remote unit of work support to access other remote databases, then each VM user machine that can be used to run QMF is known as an application requester for QMF. Each database that contains the QMF database objects is known as an application server for QMF.
- Understanding how CMS communications directories are used by DB2 for VM.

DB2 for VM requirements

QMF uses standard interfaces to the database. Because it supports only one DB2 for VM database, if you want to use QMF in more than one database, you must install QMF into each one. The QMF database installation execs prompt the installer for the name of the DB2 for VM database into which QMF is being installed. The QMF installation EXECs then issue a DB2 for VM SQLINIT command for the specified database.

A PUBLIC DBSPACE is required for saving data

A user must have a PUBLIC DBSPACE to use the QMF SAVE DATA command. The size of this DBSPACE can vary depending on user requirements.

To run the QMF Installation Verification Procedure (IVP), this DBSPACE must exist because the SAVE DATA command is used during the IVP. A minimal DB2 for VM DBSPACE (128 pages) is required to run the QMF IVP.

Database CONNECT ID "Q" and "SQLDBA"

QMF uses a CONNECT ID of Q for all control tables, sample tables, sample queries, and views. The installer does not need a VM user ID of Q; however, all installation steps that update the database issue the DB2 CONNECT command for the userid of Q.

The CONNECT ID of SQLDBA is required to set up the CONNECT ID Q. Because it was created when DB2 for VM was installed, the CONNECT ID of SQLDBA should already exist in your database.

QMF SQL install packages

During installation, QMF runs two programs that contain SQL statements. The DB2 for VM Database Utility (SQLDBSU) loads the database packages for the DSQDBINS and DSQDBSQL programs into each database server where QMF is being installed.

Further requirements

The following data base requirements exist for each database that QMF is installed in. The sections that follow describe the items in this list.

- **QMF DBSPACE requirements**

  There are ten DBSPACEs required for QMF. They are established during installation.

  QMF must have a DBSPACE to store user tables created as a result of using the QMF SAVE DATA command. You can use an existing DBSPACE or you can create a new one during the installation of QMF.

- **QMF control tables**

  There are eight QMF control tables. Each table is created in its own DBSPACE.

- **QMF catalog views**

  There are three QMF catalog views required for the QMF LIST command, enabling users to list database objects that they are authorized to use.

- **QMF sample tables**

  There are nine sample tables that are created in one DBSPACE.

- **QMF SQL packages**

QMF contains several SQL packages that must be loaded into each database into which you install QMF. The packages are loaded after the QMF control tables are created during installation.

QMF DBSPACE requirements

DB2 for VM stores tables and indexes in tables within DBSPACEs. A DBSPACE is a logical allocation of space in the database. A DBSPACE holds data in 4,096-byte blocks called pages. QMF requires the use of public DBSPACEs, which allow multiple user access at the same time; any one user can be doing update, insert, or delete functions.

Because you cannot extend DBSPACEs after they are defined, you should overestimate the required number of pages. The penalty for overestimating DBSPACE pages is nominal because the unused DBSPACE pages are not stored. On the other hand, the penalty for underestimating DBSPACE pages can be quite expensive in terms of reorganization activities required to reestablish the data in a larger DBSPACE later.

DBSPACEs must first be created and then acquired for use through the DB2 ACQUIRE DBSPACE command. Because QMF issues the ACQUIRE DBSPACE command, you must be sure you have already created the appropriate DBSPACEs.

The DBSPACEs required by QMF, as well as their contents and default sizes, are shown in Table 27.

Table 27. DBSPACEs Required by QMF

DBSPACE Name	Contents	Default Size
DSQTSCT1	Q.OBJECT_DIRECTORY table	256
DSQTSCT2	Q.OBJECT_REMARKS table	256
DSQTSCT3	Q.OBJECT_DATA table	5120
DSQTSPRO	Q.PROFILES table	128
DSQTSSYN	Q.COMMAND_SYNONYMS table	128
DSQTSLOG	Q.ERROR_LOG table	128
DSQTSGOV	Q.RESOURCE_TABLE table	128
DSQTSRDO	Q.DSQ_RESERVED table	128
DSQ2STBT	QMF sample tables	128
DSQTSDEF	QMF SAVE DATA	128

Notes:

1. The default size of these DBSPACEs may not be correct for your installation. You should evaluate the DBSPACE requirements of your installation before creating the DBSPACEs.
2. DSQTSCT3 should be your largest DBSPACE because it contains all your QMF queries, procedures, and forms. DBSPACEs DSQTSCT1 and DSQTSCT2 are created and acquired with a size of one page for each 25 pages in DBSPACE DSQTSCT3.
3. DSQTSDEF is the default name for the DBSPACE to be used by the QMF SAVE DATA command. This DBSPACE name can be changed.
4. Do not use SYS as the first three characters of a DBSPACE name; SYS denotes a DBSPACE reserved for DB2 system usage.
5. The smallest DBSPACE size that DB2 for VM allows is 128 pages. DB2 may actually give you more pages than you request because it acquires storage in units of 128 pages. DB2 determines the number of pages you receive by rounding the number you specify to the next higher multiple of 128 pages.

   Example: If you specify PAGES=53, DB2 acquires a block of 128 pages; if you specify PAGES=130, DB2 acquires 256 pages.

To determine how many of the ten DBSPACEs you need to create for your installation, perform these steps:

1. Identify the number of additional DBSPACEs that you need, based on the following considerations:
   - If you are installing QMF Version 7.2 into a database that does not contain any version of QMF, you need to create all ten DBSPACEs shown in Table 27 on page 156.
   - If you have QMF Version 3.1 or a later release installed in the same database in which you are installing QMF Version 7.2, no new DBSPACEs are needed.
2. Run the following query to list the DBSPACEs defined and their sizes. To run this query, you must have DB2 for VM DBA authority or have SELECT authority on table SYSTEM.SYSDBSPACES. Run this query using QMF or ISQL:

   ```
   SELECT * FROM SYSTEM.SYSDBSPACES
     WHERE  DBSPACETYPE=1 AND OWNER=''
   ```

   Note: If you plan to create DBSPACEs while installing QMF, see the discussion in "Step 2—Creating DB2 for VM DBSPACEs: DSQ2DBSC" on page 168. If you need to create additional DBSPACEs after QMF is installed, use the procedures described in *DB2 Server for VSE & VM Database Administration*

QMF control tables

There are eight QMF control tables, each created in its own DB2 for VM DBSPACE. (Separate DBSPACEs improves performance.) The contents of each control table are:

Table 28. The QMF control tables

Table	DB space	Contents
Q.OBJECT_DIRECTORY	DSQTSCT1	General information on all queries, forms, and procedures in the database
Q.OBJECT_REMARKS	DSQTSCT2	Comments that were saved with the queries, forms, and procedures in the database
Q.OBJECT_DATA	DSQTSCT3	Text defining the queries, forms, and procedures in the database
Q.PROFILES	DSQTSPRO	User session profiles
Q.ERROR_LOG	DSQTSLOG	Information on system, resource, and "unexpected condition" errors
Q.COMMAND_SYNONYMS	DSQTSSYN	Command synonyms
Q.RESOURCE_TABLE	DSQTSGOV	Resource and limit values for the QMF governor
Q.DSQ_RESERVED	DSQTSRDO	The information needed during QMF initialization

QMF catalog views

QMF requires the following three catalog views for the QMF LIST command and Prompted Query functions:

- Q.DSQEC_TABS_SQL is a view on the SYSTEM.SYSCATALOG and SYSTEM.SYSTABAUTH DB2 for VM system tables.
- Q.DSQEC_COLS_SQL is a view on the SYSTEM.SYSCOLUMNS and SYSTEM.SYSTABAUTH DB2 for VM system tables.
- Q.DSQEC_QMFOBJS is a view on the QMF control tables Q.OBJECT_DIRECTORY and Q.OBJECT_REMARKS.

QMF sample tables

The sample tables are placed in DBSPACE DSQ2STBT. The table contents are described in the following list. (Each table provided by QMF contains information on the fictional J & H Supply Company.)

Table Contains Information on:

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

QMF SQL packages

QMF contains SQL packages which must be loaded into each database in which QMF is installed. QMF Version 7.2 access modules contain the DSQD prefix in the SYSTEM.SYSACCESS table. For more information on access modules see *DB2 Server for VM System Administration*.

Before you begin

Before you begin installing QMF Version 7.2 review these topics.

Previous releases of QMF

If you have a previous version of QMF installed, you can install the new release of QMF into a different DB2 for VM database for testing purposes, or you can install and run both releases in the same database concurrently. If you install QMF Version 7.2 in the same database as the previous release, make certain that the sample tables of the previous release are not used during installation.

Migration and fallback

Note: Skip this section if QMF is being installed for the first time.

Your users might need help before they can operate the new release of QMF. Supplying this help is what "migration" means.

If you decide to return to your earlier release of QMF, your Version 7.2 users might need help. Supplying this help is what "fallback" means.

Migration and fallback are post-installation operations. They are described in Appendix C, "Migration and Fallback". For planning purposes, you should read about them before beginning the Version 7.2 installation.

QMF National Language Feature (NLF) considerations

The QMF National Language Feature (NLF) is a software feature that provides QMF users with a QMF environment tailored to the language of their choice. NLFs enable users to enter QMF commands, view help, and perform QMF tasks in languages other than English. NLFs are installed as separate features of QMF.

Example

When a user elects to operate QMF in a German-language environment, QMF commands, keywords, panels, and messages are displayed in German.

A NLF does not provide any new QMF function. In general, anything users can do in the base English-language session can be done in an NLF session, and vice versa. For the most part, 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**.

A QMF NLF is installed *after* you have installed QMF. For a description of NLF, see Chapter 15, "Installing a QMF Version 7.2 National Language Feature (NLF)", on page 185.

Some names of programs and phases shown in this book have an *n* symbol in them, indicating that the name can vary. If you are using an NLF, replace all *n* symbols you see in this book with the one-character national language identifier (NLID) from Table 29 that matches the NLF you installed. The table also shows the names by which QMF recognizes each language.

Table 29. NLIDs representing QMF base (English) and National Language Features

NLF	NLID	Name 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

Table 29. NLIDs representing QMF base (English) and National Language Features (continued)

NLF	NLID	Name QMF uses for this NLF
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.

installing QMF into a workstation database server on VM

In order to access remote database servers from QMF on VM, DRDA APPC communications must be in place between VM and the remote server. VM uses VTAM and AVS definitions for the remote server. These definitions are accessed via the CMS COMDIR NAMES file, in which the VM gateway, DB2 remote server name, mode name, and session limits are defined for the remote DRDA connection.

In addition, you must have a database created on the workstation database server and you must have SYSADM authority to that database for your install ID.

Some QMF install steps use the SQLDBSU DB2 for VM utility. Prior to running the QMF installation EXEC (DSQ2EINS), you must install SQLDBSU into the remote database server.

For more information about installing SQLDBSU into a remote database server, see *DB2 Server for VSE & VM Database Services Utility for IBM VM Systems*

Chapter 14. Installing QMF Version 7.2 into the DB2 for VM Database

This chapter explains the steps for performing the installation of QMF Version 7.2 from VM. If you have already installed QMF Version 7.2 and want to install it into another database, follow the directions in this chapter.

If you are installing QMF Version 7.2 for the first time, read the *QMF Program Directory* first and complete the steps listed therein to unload QMF from tape to disk. Check the program directory for modifications to the procedures described in this chapter, then complete the steps in this chapter to complete the QMF database installation.

The QMF installation uses the Restructured Extended Executor (REXX) language execs to install QMF into a DB2 database. For information on how to use REXX, see the *VM System Product Interpreter Reference* manual.

Installation considerations:

1. The QMF-supplied execs that install QMF into a database are designed to prompt the installer for variable information. There is no requirement for your installation to change the supplied installation execs. Every prompt message asks for variable input, and each offers an optional help or cancel response.
   - If help is issued, a small abstract of the prompt request is displayed.
   - If cancel is issued, the exec terminates.
2. All variables are resolved before execution of any given installation step, which can be restarted from the beginning.
3. Several output files from the execs are routed to the printer. You may want to spool your printer to HOLD before you start the database installation.

QMF installation flow diagram

Figure 33 on page 164 is a flow diagram of QMF installation to help acquaint you with the installation process before starting.

Installing QMF 7.1

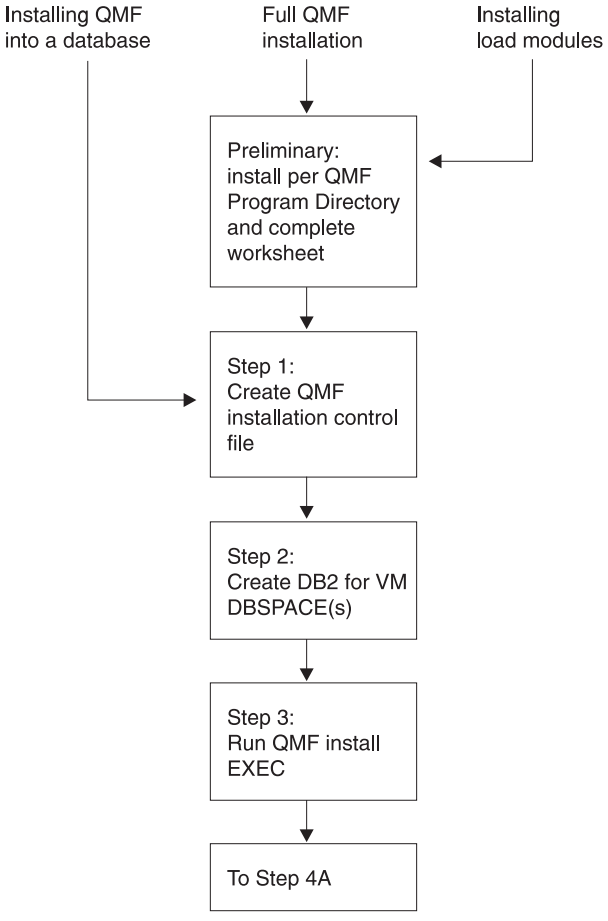


Figure 33. Installation steps for QMF Version 7.2 (Part 1 of 2)

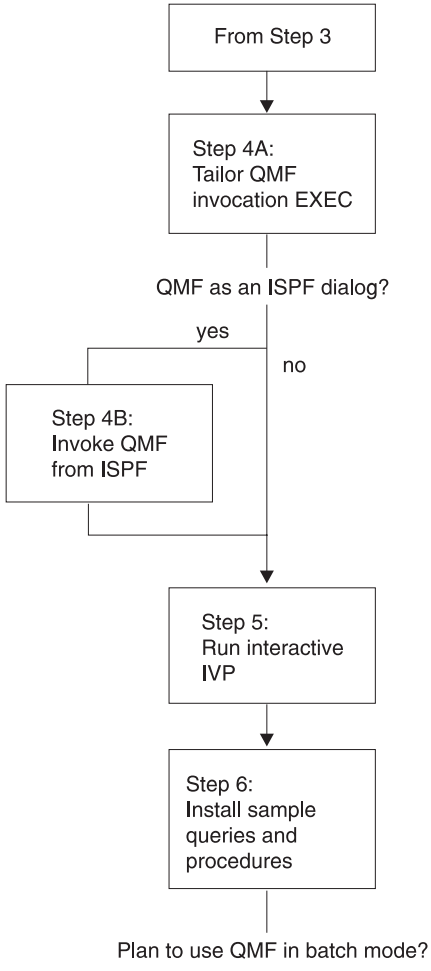


Figure 34. Installation steps for QMF 7.2 (Part 2 of 2)

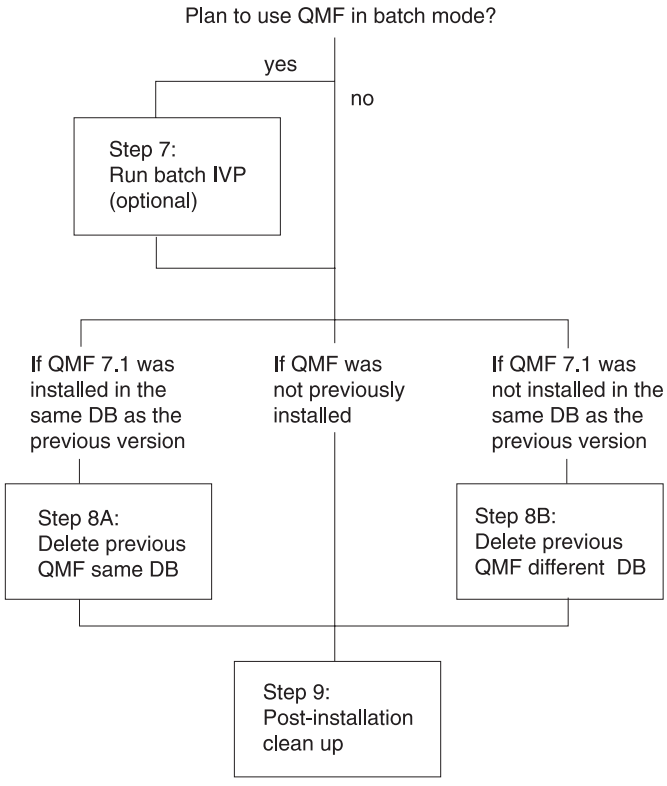


Figure 35. Installation steps for QMF 7.2 (Part 3 of 3)

The installation steps

The installation steps are outlined on the following pages.

If you are performing a QMF Version 7.2 migration installation, that is, if you are installing QMF Version 7.2 into a database that already has a previous level of QMF installed, follow all the installation steps, indicating the previous QMF level when required.

Preliminary: read the program directory and complete the QMF Version 7.2 worksheet

Before beginning the installation process, read the *QMF Program Directory* shipped with the ISD tape for supplementary data. The program directory contains all steps for installing QMF from tape to disk and building the DCSS. You must complete the steps in the program directory before doing the installation steps in this book. Only QMF installation from DB2 for VM is described in this book.

The following worksheet lists the information you provide during QMF installation.

Table 30. Information Required during QMF Installation (QMF 7.2 Worksheet)

Information required for:	Supply data fields containing _____			
	No prior QMF	QMF Migration	QMF in DB2 Workstation Server	QMF in iSeries Server
• Database/location name	_____	_____	_____	_____
• Database type (DB2 VM, DB2 Workstation Server, or DB2 UDB for iSeries)	DB2VM	DB2VM	DB2WS	DB2400
• Prior QMF Version/Release level (if any)	N/A	_____	N/A	N/A
• SQLDBA CONNECT password	_____	_____	N/A	N/A
• Q CONNECT password	_____	_____	N/A	N/A
• Default DBSPACE name for SAVE DATA command (default is DSQTSDEF)	_____	_____	N/A	N/A
• Number of DBSPACE pages for: **DBSPACE NAME (default)**				
Q.OBJECT_DATA table (5120)	____	N/A	N/A	N/A
Q.PROFILES table (128)	____	N/A	N/A	N/A
Q.ERROR_LOG table (128)	____	N/A	N/A	N/A
Q.COMMAND_SYNONYMS table (128)	____	N/A	N/A	N/A
Q.RESOURCE_TABLEtable (128)	____	N/A	N/A	N/A
SAVE DATA command (128)	____	N/A	N/A	N/A

Use DB2WS as the database type for all workstation database servers. Use DB2400 as the database type for DB2 iSeries database servers.

The QMF table spaces created in workstation database servers are system-managed. Thus, they have no default size.

Step 1—Create QMF installation control file: DSQ2ECTL

The QMF exec, DSQ2ECTL, prompts you for information that is required in the QMF installation process.

To create the QMF installation control file, do the following:
1. Access the QMF distribution disk in WRITE mode.
2. Ensure that your A disk has enough room to generate temporary files.
3. Supply the information for the worksheet, if you have not yet done so.
4. Run the exec: DSQ2ECTL.

Prompts

You receive a series of prompts that request the information you developed using the worksheet (Table 30 on page 167).

Anytime during this process, you can enter:
- `HELP` on the command line for information
- `CANCEL` to terminate the process before completion

A file, QMFV720E INSTALL, is created on your installation disk. It contains the information you supplied to the previous prompts.

If an installation file already exists from a previous installation, the information you enter is appended to this file. The previous information is "deactivated", but saved, for service purposes.

Step 2—Creating DB2 for VM DBSPACEs: DSQ2DBSC

Note: Skip this step if one or more of the following are true:
- You are installing QMF into a remote database server.
- The database you are installing QMF Version 7.2 into has QMF Version 3.1 or later already installed.
- There are sufficient public DBSPACEs available for the DB2 for VM database of the sizes indicated in the installation worksheet (Table 30 on page 167). You can check this by invoking ISQL and issuing the following:

```
SELECT * FROM SYSTEM.SYSDBSPACES
  WHERE  DBSPACETYPE=1 AND OWNER=''
```

To create the DBSPACEs required by QMF, do the following:
1. Access the QMF distribution and production disks.
2. Ensure that the QMF installation control file QMFV720E INSTALL exists on the distribution disk.
3. Ensure that you have an A-disk to generate a temporary file.
4. Run the exec: DSQ2DBSC.

   This exec will:
   - Use the QMFV720E INSTALL file on the QMF distribution disk to determine whether or not this is a new or migration install. If this is a

new install, all ten DBSPACEs are created. If this is a migration from QMF V2R4 or an earlier release, only one DBSPACE is created.

- Prompt you to enter the storage subpool you want to use.
- Create the 'dbname SQLADBSP A' file ('resid SQLADBSP A' file if the database you are installing QMF into is Version 7.2) on your A-disk. ('dbname' is the database name and 'resid' is the resource ID for your DB2 for VM database.)

5. Send the 'dbname SQLADBSP ' file (or 'resid SQLADBSP') file to the database virtual machine.
6. Log onto the database virtual machine and stop the database. (Typically with the SQLEND command.)
7. Receive the 'dbname SQLADBSP' (or 'resid SQLADBSP') file to the A-disk.
8. Access the DB2 for VM service disk (DASD 193) as the V-disk.
9. Run the SQLADBSP exec, by entering:

   ```
   SQLADBSP DB(dbname)
   ```

   where dbname is the name of the DB2 for VM database. DBSPACE(s) is added based on the information in the dbname SQLADBSP file.

   You receive the following message:

   ```
   dbname SQLADBSP WAS FOUND.
   SHOULD THIS FILE BE USED FOR ADD DBSPACE?
   ```

   Answer YES.

   You receive a message inquiring whether or not you want to modify the dbname SQLADBSP file.
   - To edit the file, answer YES.
   - To continue without editing, answer NO.
10. Release the DB2 for VM service disk (DASD 193).
11. Restart the database and continue with the installation, by entering:

    ```
    SQLSTART DB(dbname)
    ```

    where dbname is the name of the DB2 for VM database.
12. Run the following query using ISQL to verify that the new DBSPACEs are available for QMF:

    ```
    SELECT * FROM SYSTEM.SYSDBSPACES
      WHERE  DBSPACETYPE=1 AND OWNER=''
    ```

    To run this query, as a minimum you need to have SELECT authority on table SYSTEM.SYSDBSPACES, or have DB2 DBA authority, which implies the SELECT privilege.

Step 3—Run QMF installation exec: DSQ2EINS

This section describes the following topics:
- Preparing to run the installation exec
- What the installation exec does
- Running the installation exec
- Installation exec error messages

Preparation

The information you provided in Step 1—Create QMF installation control file: DSQ2ECTL is used by the QMF installation exec. Before running this exec:

1. You must have access to the QMF distribution disk in WRITE mode.
2. Ensure that the QMF installation control file QMF720E INSTALL exists on the distribution disk.
3. Ensure that you are linked to the DB2 for VM production minidisk in READ mode.
4. You can let the printer and console continue processing unless a severe error is found, by issuing the following CMS commands:

```
spool prt cont hold
spool console start cont
```

Assumptions for installing QMF into a remote database server

Before you attempt to install QMF on a remote database server, be sure to complete the necessary pre-requisites described in "installing QMF into a workstation database server on VM" on page 161.

What the installation exec does

All output from the installation exec is routed to the virtual printer spool file.

Substeps:
- Substep 3.1: Builds the SQL commands to acquire the DB2 DBSPACEs.
- Substep 3.2: Establishes a DB2 for VM CONNECT ID of Q.
- Substep 3.3: Reloads the QMF installation program packages.
- Substep 3.4: Creates the QMF control tables and QMF catalog views.
- Substep 3.5: Reloads the QMF SQL Packages into a DB2 for VM database.
- Substep 3.6: Discards any QMF sample tables, if they exist.
- Substep 3.7: Creates the QMF Version 7.2 sample tables.

Running the QMF installation exec

To start the installation exec, issue:

```
DSQ2EINS
```

Restart procedure: If this exec fails, use the following procedure to restart the exec and continue where you left off:

1. Determine what the problem is and fix it.
2. Rerun this exec with an input parameter equal to the restart value provided in the message after the exec terminates.

For example, if you receive the message:

```
TERMINATING EXECUTION ...
TO RESTART THIS EXEC AND CONTINUE WHERE YOU LEFT OFF:
  - FIX THE PROBLEM ENCOUNTERED.
  - RERUN THIS EXEC WITH THE INPUT PARAMETER OF 2
```

You can restart the exec with the statement:

```
DSQ2EINS 2
```

Installation exec error messages

If you encounter a problem running the QMF installation exec, you need to find the error message describing the problem. This error message may be sent to either the console or the printer; therefore you may want to spool your console and your printer to "HOLD".

If you choose to spool your printer or console, be aware that you may have to enter both of the following statements to release the file that contains the error information:

```
spool prt close
spool console close
```

Error messages produced by the SQLDBSU exec are sent to the printer. If you see a console message like "Errors processing SQLDBSU", you should examine the output sent to the printer. The command to transfer the printer files to your reader, so that you can view them there, is:

```
TRANS PRT ALL *
```

Look in the *DB2 Server for VM Message and Codes* manual for explanations of any error messages starting with "ARI".

Step 4—Start QMF: DSQ2EINV

This section describes tailoring the QMF invocation EXEC and establishing QMF as an ISPF dialog (optional).

Step 4A—Tailor the QMF invocation exec: DSQ2EINV (optional)

The sample QMF invocation exec, located on the production minidisk, is executed when a user wants to invoke QMF interactively in the VM environment. The first part of the exec, DSQ2EINV, is shown in Figure 36 on page 173. Modify only the indicated variables to tailor the exec for your installation.

Using DSQQMFE and ISPSTART: The parameter values that exist in DSQ2EINV are used unless you specify different values when you invoke the exec. You can do this through DSQQMFE or the ISPSTART command. Parameters values specified in this way override those set in the QMF callable interface REXX exec DSQSCMDE, which is on the production minidisk.

Note: DSQ2EINV is only a sample QMF invocation exec. The necessary links to minidisks, FILEDEFs, SQLINIT, and ISPSTART commands are described clearly in simpler QMF invocation exec. These execs, DSQ2EIN1 (with ISPF) and DSQ2EIN2 (without ISPF), are located on the production minidisk. You may find them useful in constructing your own QMF invocation exec to match your environment requirements.

For clarification of ISPF files, see *ISPF for VM Dialog Management Services and Examples*

```
/*-------------------------------------------------------------------*
 *                                                                   *
 *  Sample QMF invocation EXEC                                       *
 *                                                                   *
 *  EXEC NAME:         DSQ2EINV EXEC                                 *
 *                                                                   *
 *  Status: Version 7 Release 2 LeveL 0                              *
 *                                                                   *
 *  Input:    DB(dbname)        - optional, default 'SQLDBA'         *
 *            PGM(program)      - optional, default 'DSQQMFE'        *
 *            MODE(runmode)     - optional, default 'I'              *
 *            PROC(procedure)   - optional, no default               *
 *            CMSSUB(subset_restriction) - optional, default 'YES'   *
 *            ISPF(use_ispf)    - optional, default 'YES'            *
 *                                                                   *
 *  Note:  If you have any level of DB2 VM, GDDM, ISPF, QMF or       *
 *         QMF NLF already attached when you execute this exec,      *
 *         the corresponding disk in this exec will not be linked,   *
 *         and the existing disk will be used.                       *
 *                                                                   *
 *-------------------------------------------------------------------*/

 parse upper arg parm1 parm2 parm3 parm4 parm5 parm6 junk

 lchar = 'E'                    /* QMF language feature identifier */

/*-------------------------------------------------------------------*
 *  The following are the variables which may need to be tailored    *
 *  for your installation.                                           *
 *  Note:  If you are using SFS directories, replace the link        *
 *         information with 'FILEPOOL:USERID.DIRNAME'.               *
 *-------------------------------------------------------------------*/

 dcssname = 'QMF720'||lchar      /* QMF DCSS name for ISPSTART      */
 sql_link = 'SQLMACH 195 195'    /* DB2 VM minidisk link information*/
 qmf_link = 'P697F4BA 400 400'   /* QMF Production minidisk         */
                                 /* link information                */
 dbname = 'SQLDBA'               /* set default database name       */
 program = 'DSQQMF'||lchar       /* set default QMF program name    */
 mode = 'I'                      /* set default QMF run mode        */
 procedure = ''                  /* no default procedure            */
 subset = 'YES'                /* default to CMS subset restrictions*/
 ispf = 'YES'                  /* link to ISPF minidisk (optional)  */

/*-------------------------------------------------------------------*
 *               END OF TAILORABLE VARIABLES                         *
 *-------------------------------------------------------------------*/
```

Figure 36. Sample QMF Invocation exec (DSQ2EINV)

Notes on Figure 36:

1. The correspondence between the variables on the sample exec and the parameters on the ISPSTART command is as follows:

a. **PGM** is used as the **PGM** parameter on ISPSTART.
b. **MODE** is used as the **DSQSMODE(M)** parameter on ISPSTART.
c. **PROC** is used as the **DSQSRUN(I)** parameter on ISPSTART.

2. If you specify 'NO' for the ISPF parameter, the CMSSUB parameter is ignored.

   If you specify 'YES' for the ISPF parameter or take the default (YES), either of the following happens:
   - If CMSSUB = NO, then ISPF is started via SELECT DCSS.
   - If CMSSUB = YES, then ISPF is started via SELECT PGM.

   When ISPF executes a SELECT PGM, the ISPF product turns on the CMS SUBSET indicator, whereas if ISPF executes a SELECT DCSS, the ISPF product does not turn on the indicator.

3. Following are examples of invocation statements:
   - `DSQ2EINV MODE(I)`

     This statement invokes QMF interactively. (It is normally the default.)
   - `DSQ2EINV MODE(B) PROC(MYPROC)`

     This statement runs the procedure MYPROC in batch mode.

QMF dialog considerations: The following considerations apply to the QMF dialog:

- Virtual Machine considerations

  The virtual machine size should be at least 5.0 MB of storage without ISPF or 6.0 MB with ISPF. If a larger virtual machine size is available, QMF uses it when the user scrolls through a report. QMF requires that both ISPF (if used) and DB2 for VM be running in disconnected virtual machines before it can be invoked.

- Program modules

  Before you invoke QMF, the DB2 for VM database, QMF's discontiguous shared segments, ISPF's shared segments (if used), and GDDM's shared segments or product text libraries must be available.

- QMF data files

  The following list describes the files used by QMF. These files are allocated according to the recommended sizes in the DSQ2EINV exec. If you want to allocate them differently, you must modify the invocation exec.

  – DSQDEBUG—QMF trace dump output

    If the trace option is set to trace during initialization or during a QMF session, QMF's trace output is used. It is also used if QMF abnormally terminates. This file must be allocated prior to invoking the QMF dialog.

    The trace output is formatted in two different formats on the basis of the allocated record size. If the record is greater than 120, the output is generated in eight fullword columns; otherwise, the output is generated

in four fullword columns appropriate for viewing on a terminal. The record format RECFM can be fixed or variable, with a block size that is a multiple of the record size.

– DSQPRINT—Print data output

  The print data output contains print data that is produced by a QMF PRINT command issued during a QMF session. This file can be allocated by using the QMF CMS command while the QMF dialog is running or it can be allocated prior to invoking the QMF dialog.

  RECFM can be FBA or VBA. It is recommended that this file be allocated with a record length (LRECL) supported by your printer device type.

– DSQSPILL—Spill data file

  The spill file is used when QMF runs short of virtual storage when producing data for a report that is requested during a QMF session. This file can be allocated by using the QMF CMS command to invoke the CMS FILEDEF command, while the QMF dialog is running or it can be allocated prior to invoking the QMF dialog. The spill file is a fixed unblocked file with a record length (LRECL) of 4,096.

  Note: The larger the user's spill file, the less often the user encounters the "incomplete data" condition.

– DSQEDIT—Edit transfer file

  This file is used whenever a QMF EDIT command is issued during a QMF session. This file is a fixed record file with a record length (LRECL) of 79.

– DSQPNLE—QMF panel file

  This file contains all the QMF panel definitions. It is created during QMF installation.

– DSQLDLIB—QMF load library

  This file must be allocated to ISPLLIB and globally defined.

QMF-GDDM considerations: When the QMF DCSS is built, it includes the GDDM interface code. If you run GDDM from a DCSS, you need not access a GDDM disk, or GDDM TXTLIBs, and you may remove the lines in the invocation EXEC that refer to GDDM.

However, if you do not have GDDM in a DCSS, you must access the GDDM TXTLIBSs and perform the necessary FILEDEFs. If you want to change the release of GDDM being used by QMF, you must rebuild the QMF DCSS. See the *Program Directory* for information on building the QMF DCSS.

QMF-DB2 for VM considerations include the following:

- QMF supports DATE, TIME, and TIMESTAMP data types. So users can make use of local date/time exit routines.

When planning for local date/time exit routines, it is important to keep in mind that these are DB2 for VM exits, they are not QMF exits. For details about how these exits are created, refer to *DB2 Server for VM System Administration* manual.

In order for QMF to use a local date/time exit, the text files containing the date/time exits ARIUXDT and ARIUXTM must be placed on a minidisk that is accessible to QMF when QMF starts.

If QMF is being started by DCSS mode, two relocatable module files must be created from the existing exit text files ARIUXDT and ARIUXTM. To create the relocatable module files issue the following CMS commands:

```
LOAD    ARIUXDT ( RLDSAVE )
GENMOD  ARIUXDT
LOAD    ARIUXTM ( RLDSAVE )
GENMOD  ARIUXTM
```

- The QMF DCSS includes the ARIRVSTC text file, and if this file is changed by PTFs applied to DB2 for VM or a new level of DB2 for VM, the QMF DCSS must be re-built. See the *Program Directory*.

QMF-DXT considerations: If you want to start Data Extract (DXT) from QMF, the ISPF setup for DXT should be merged with the ISPF setup of QMF. You can do this in either of the following ways:

- Combining the QMF and DXT ISPF library FILEDEFs (concatenating the MACLIBs under the same ISPF ddname). Give some thought to how you want the libraries concatenated. If QMF is generally used more than DXT, its libraries should be concatenated ahead of DXT's.
- Using the ISPF LIBDEF service to dynamically allocate DXT's libraries under QMF. This can be done in lieu of, or in addition to, the merging of the ISPF setups.

QMF provides a sample exec, DSQABX2L, which contains an example of how to use LIBDEF for DXT.

Step 4B—Invoke QMF from an ISPF environment (optional)

ISPF supplies a Master Application Menu as part of its installation process. The QMF dialog can be invoked from the ISPF Master Application Menu, or any other selection menu that you want to use. For an example of how the ISPF Master Application Menu appears after adding QMF, see Figure 37 on page 177.

The ISPF LIBDEF service provides applications with a dynamic method of defining application data elements files while in an active ISPF session. For more on the ISPF LIBDEF service, see *ISPF for VM Dialog Management Services and Examples*.

```
%------------------------  MASTER APPLICATION MENU  --------------------
%SELECT APPLICATION ===>_OPT     +
%                                                          +USERID   -
%                                                          +TIME     -
%   1 +SPF        - SPF PROGRAM DEVELOPMENT FACILITY       +TERMINAL -
%   2 +QMF        - QMF QUERY MANAGEMENT FACILITY          +PF KEYS  -
%   3 +QMF2       - QMF with alternate database
%
%
%
%
%
%
%
%
%
%   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(ISP@PRIM) NEWAPPL'
                2,'CMD(DSQ2EINV)'                          (Note 1)
                3,'CMD(DSQ2EINV DB(SQLDBA2))'              (Note 2)
       /*                          */
            /* ADD OTHER APPLICATIONS HERE */
            /*                          */
                P,'PANEL(ISPOPT)'
                X,'EXIT'
              ' ',' '
                *,'?' )
)END
```

Figure 37. QMF dialog on ISPF master application menu

Note: The default database name is SQLDBA. You can modify the name within the QMF invocation exec DSQ2EINV, to suit your installation. You can specify another database name as a parameter.

Step 5—Running IVP for QMF interactive mode : DSQ2EIVP

Note: Be sure that you have installed QMF Version 7.2 in the database you are going to use, and that you are connected to that database.

The Installation Verification Procedure (IVP) session tests the following:

1. Initialization for a QMF session
2. The existence of QMF help panels
3. Importing of the initial IVP procedure
4. The existence of QMF control tables
5. The operation of the QMF database modules
6. The created table through the SAVE DATA command
7. The operation of QMF PRINT, EXPORT, IMPORT, and CMS commands
8. The trace facility
9. The QMF command interface

The IVP procedures are in the sample files on the production minidisk.

As a result of the IVP:
- A query is printed.
- A trace is saved in a file named DSQDEBUG.
- A query is exported to a file named QMFIVP QUERY A1.
- A query is imported from a file named QMFIVP QUERY A1.
- The file QMFIVP QUERY A1 is erased, using the CMS command.

Step 5A—Test QMF initialization
To run the IVP, first get to the QMF Home Panel using the DSQ2EINV sample invocation exec or your own QMF invocation exec.

During the IVP, you might get QMF error messages; if you do, press the Help key to get additional information.

Step 5B—Test the Help panel
When you have successfully initialized QMF, test for the Help panel. To do this, press the Help key from the home panel. After you are on the Help panel, press the Exit key to take you back to the home panel.

Step 5C—Test the QMF command interface (ISPF only)
To test the QMF command interface, issue the following command:

```
CMS DSQ2ECI1
```

If this exec runs successfully, your QMF profile is displayed and you receive a confirming message.

Check your profile for the correct values. For example, verify that the DBSPACE value matches what you specified during “Step 1—Create QMF installation control file: DSQ2ECTL” on page 167. If the DBSPACE value is not correct, update your profile to contain the correct value before you continue.

Step 5D—Testing the QMF IVP procedure
If you are running the IVP against QMF installed on a DB2 for VM server, issue the command:

```
CONNECT Q (PASSWORD=xxx
```

where “xxx” is the value given to the Q CONNECT password when the QMF installation control file is built.

If you are running the IVP to any database other than DB2 for VM, the connect ID at that server must have DBA or SYSADM authority.

Next, issue the command:

```
IMPORT PROC FROM DSQ2EIVP PROC *
```

Now press the Run key or issue the RUN PROC command to run the procedure. Answer YES to all prompts. If the procedure runs successfully, you get a message indicating this. If the procedure does not run successfully, determine the problem by using the QMF messages and by pressing the Help key to see the message help panels.

Restarting the IVP

The IVP can be restarted from the beginning at any time by importing and running the starting QMF procedure. Follow the procedures from the beginning of this step.

Step 6—Installing QMF sample objects and application objects: DSQ2ESQD and DSQ2ESQI

After QMF is installed and tested, you can use it to import the sample queries (all saved with SHARE='YES' option), batch IVP procedures, and sample applications. The QMF procedure and queries used to import the sample queries are on the QMF distribution minidisk (documented in the *Program Directory*).

If you have a previous version of QMF installed, you must delete those sample queries and procedures before installing QMF Version 7.2 queries and procedures.

Perform the following steps to install the sample queries and procedures:

1. Start QMF if not already logged on from "Step 5—Running IVP for QMF interactive mode : DSQ2EIVP" on page 177.
2. If not done in "Step 5—Running IVP for QMF interactive mode : DSQ2EIVP" on page 177, and you are installing on a DB2 for VM server, issue the command:

   ```
   CONNECT Q (PASSWORD=xxx
   ```

   where xxx is the password of Q.

   If you are running these jobs to any database other than DB2 for VM, the connect ID at that server must have DBA or SYSADM authority.
3. If you have a previous version of QMF installed, delete previous sample queries and procedures by importing and running procedure DSQ2ESQD, as follows:

   ```
   IMPORT PROC FROM DSQ2ESQD PROC *
   ```

   Press the Run key or issue the RUN PROC command.
4. Install sample queries and procedures by importing and running procedure DSQ2ESQI, as follows:

   ```
   IMPORT PROC FROM DSQ2ESQI PROC *
   ```

Press the Run key or issue the RUN PROC command.

Restarting the procedure

If a failure occurs during this procedure, correct the error, then run procedure DSQ2ESQD to delete any previously created sample queries. Rerun procedure DSQ2ESQI to install sample queries and procedures.

Step 7—Run the batch-mode IVP (optional): DSQ2EBAT

If you plan to run QMF procedures in batch mode, run this IVP to ensure that QMF for batch mode processing has been successfully installed. The same files and DB2 for VM authorization used in QMF interactive mode are also required to run QMF procedures in batch mode.

The Installation Verification Procedure (IVP) tests the following batch-mode operations:

1. Reaching and initializing QMF
2. The existence of QMF control tables
3. The operation of the QMF database modules and issuing the SAVE DATA command
4. The operation of the QMF PRINT, EXPORT, IMPORT, and CMS commands and the trace facility

The IVP procedures are in the sample files on the QMF distribution disk.

Your CMS PROFILE EXEC should define a print file (DSQPRINT) and a message file (DSQDEBUG).

As a result of the IVP:

- A query is printed.
- A trace file is saved in a file DSQDEBUG.
- A query is exported to file "QMFIVP QUERY A1".
- A query is imported from file "QMFIVP QUERY A1".
- File "QMFIVP QUERY A1" is erased using the CMS command.

During the IVP, you might receive QMF error messages. For the text of the error messages, see the DSQDEBUG file. For more information on these error messages, you can use the QMF HELP command to view the message help panels.

DB2 for VM authorization

If you (the installer) do not have DB2 for VM DBA authority or an authorization ID of "Q", the minimum DB2 for VM authorization required is:

- SELECT authority for all QMF control tables. The following are examples of SQL GRANT statements to give SELECT authority for Q.PROFILES and Q.ERROR_LOG:

```
GRANT SELECT ON Q.PROFILES TO installerid
GRANT SELECT ON Q.ERROR_LOG TO installerid
GRANT RESOURCE TO installerid
```

- DELETE and UPDATE authority for Q.OBJECT tables. The following are examples of SQL GRANT statements to give all authority to the Q.OBJECT tables:

```
GRANT ALL ON Q.OBJECT_DIRECTORY TO installerid
GRANT ALL ON Q.OBJECT_DATA TO installerid
GRANT ALL ON Q.OBJECT_REMARKS TO installerid
```

To run the batch mode IVP, use the QMF invocation exec specifying the parameters for batch mode and the QMF procedure Q.DSQ2EBAT:

```
DSQ2EINV MODE(B) PROC(Q.DSQ2EBAT)
      or
DSQ2EINV MODE(B) PROC(Q.DSQ2EBAT) CMSSUB(NO)
```

If QMF was not installed correctly, QMF does not initialize, and you receive error messages. For the text of the error messages, see the DSQDEBUG file. For more information on these error messages, you can use the QMF HELP command to view the message help panels.

Restarting the batch IVP

This IVP starts the DSQ2EINV exec with the appropriate parameters.

Expected results from executing the batch IVP

The output looks like the following example. (The dots indicate the beginning and ending of trace records.)

```
------------------------------------------------------------
YOU MAY ENTER A COMMAND.
------------------------------------------------------------
RUN PROC Q.DSQ2EBAT
------------------------------------------------------------
SET (CONFIRM=NO)
SET PERFORMED. PLEASE PROCEED.
.
.
.
SAVE DATA AS QMF_IVPDATA
DATA WAS SAVED AS QMF_IVPDATA IN THE DATABASE.
.
.
.
OK, YOUR PROCEDURE WAS RUN.
------------------------------------------------------------
EXIT       THE EXIT COMMAND TERMINATES QMF
```

Step 8—Deleting previous versions of QMF (optional): DSQ2BDEL

Attention: Do not run this step unless you have successfully completed the installation and testing of QMF Version 7.2 and no longer need the previous release.

Optionally, run the DSQ2BDEL exec to delete a previous version of QMF. The DSQ2BDEL exec prompts for all necessary information needed to delete QMF.

Confirmation of the deletion is required before the actual deletion is done. You must be linked to the QMF distribution disk and the DB2 VM production disk, the DB2 database machine must be active, you must have DRDA connectivity to the target database, SQLDBSU must be installed in the target database, and you must have authority to perform the database deletes. There are two types of QMF deletions as defined below.

- If you have an earlier release of QMF installed in the same database in which you have installed QMF Version 7.2, run DSQ2BDEL exec with the PACKAGE option to delete the QMF database access modules of the prior release.
- If you have an earlier release of QMF installed in a different database from where you have installed QMF Version 7.2, run DSQ2BDEL exec with the FULL option to drop ALL QMF DBSPACEs in addition to the database access modules (packages) of the prior release.

Step 9—Post-installation cleanup

The QMF installation control file QMFV720E INSTALL resides on your QMF distribution disk and contains the DB2 for VM CONNECT passwords for SQLDBA and Q. This is a security exposure and should be corrected as soon as possible. You can edit the installation control file and blank out the password values. You may wish to change the DB2 for VM CONNECT password for Q and/or REVOKE DBA authority from Q, especially if you have chosen a non-trivial password for Q during QMF installation.

QMF uses the PROTOCOL (AUTO) option to run SQLINIT exec. If the PROTOCOL (AUTO) option is not used at your machine, run SQLINIT to change the default PROTOCOL.

On the CMS command line, enter:

```
SQLINIT PROTOCOL(protocol)
```

where *protocol* is SQLDS, AUTO, or DRDA.

Step 10—Load QMF database packages to a remote server (optional): DSQ2BPKB

In order for a QMF Version 7.2 requester installation to be able to communicate to a server, QMF Version 7.2 packages must be present at the server. If a complete QMF Version 7.2 new or migration installation was performed at the server, communications can be started and nothing further needs to be done.

For those servers containing QMF Version 3.2 or above where migration is not an option, you can run the install package job, DSQ2BPKB, to install QMF Version 7.2 packages at the remote server. Then access from QMF for VM Version 7.2 to that remote server is enabled. Following is a list of the DB2

servers types that are supported from QMF for VM for remote access and the minimum version/release required at the server.

- DB2 for OS/390 Version 3.1
- DB2 for VM/VSE Version 3.5
- DB2 Universal Database Version 5
- DataJoiner Version 2
- DB2 Common Server Version 2.1
- DB2 Parallel Edition Version 1.2
- DataJoiner Version 1.2
- DB2 UDB for iSeries Version 4.4

Here is a list of considerations for running the job (DSQ2BPKB) to load QMF database packages to a remote server:

1. The application server must contain at least QMF Version 3.2. For brand new installs, the QMF installation package and QMF control tables (at least) must be present.
2. DRDA communications between the DB2 application requester and the DB2 application server must be defined and operational.
3. The DB2 DRDA application server must be started.
4. The connect userid at the server must have administrator authority.
5. This job can be rerun.

Step 11—Recreate QMF views (optional): DSQ2BVW

There may be times when it is necessary to recreate the QMF control tables views. This can be accomplished by running the job DSQ2BVW, which is on the QMF distribution disk. This job will DROP and CREATE al QMF control table views and perform the necessary GRANTs to any QMF supported DB2 server. The following prerequisites must be met before the job can be run:

- QMF Version 7.2 must be installed on the target server.
- You must be linked to the QMF VM distribution disk.
- You must be linked to the DB2 VM product disk, and the DB2 database machine must be active.
- DRDA connectivity to the target server must be configured.
- The CONNECT ID at the target server must have the following authorities:
  - DBA authority on DB2 VM and VSE
  - SYSADM authority on DB2 OS/390
  - SYSADM authority on DB2 workstation
  - *ALLOBJ authority on iSeries

On the CMS command line, enter:

```
DSQ2BVW dbn
```

where dbn is the name of the target DB2 database server that you want to recreate the QMF control table views on. For further details on the DSQ2BW job, see “QMF views” on page 733

Note: This job can be rerun from the beginning.

Chapter 15. Installing a QMF Version 7.2 National Language Feature (NLF)

This chapter parallels the installation steps for QMF Version 7.2. Where there are significant procedural differences, this chapter explains the procedures to follow when installing the National Language Feature. Where the job, library, or program name differs, this chapter provides the proper names, but the procedures you follow are in Chapter 14, "Installing QMF Version 7.2 into the DB2 for VM Database", on page 163.

NLF installation execs

The QMF product ships CMS execs written in the Restructured Extended Executor (REXX) language. The execs and control statements for each NLF are shipped on the ISD tape for that feature. For information on how to use REXX, see *Virtual Machine/System Product Interpreter User's Guide*

The QMF NLF installation execs are designed to prompt the installer for variable information. There is no requirement for your installation to change the supplied installation execs. Every prompt message asks for variable input, and each offers an optional Help or Cancel response.

- When Help is issued, a small abstract of the prompt request is displayed.
- When Cancel is issued, the exec stops.

Whenever a module, library, or job named in this chapter contains the letter *n*, replace the *n* with the appropriate letter for the national language you are installing. See your *QMF NLF Program Directory* or "QMF National Language Feature (NLF) considerations" on page 160 for the appropriate letter to use for your installation.

Installing a National Language Feature

When you install an NLF, a row is added to the QMF profile table (Q.PROFILES) to support the language. This row is inserted with a user ID of SYSTEM. A unique row is added for each language that you install.

The NLF must be installed in each database you want to use it in. If you are installing into a database that contains a prior release of QMF NLF, ensure that the sample tables and views of the prior release are not used during the installation process.

You use your national language for the QMF commands to import, export, and run some installation procedures. See the NLF program directory for a list of the translated books (the translated books should have the translated QMF commands).

Hardware and program product requirements

Make sure that your GDDM and ISPF (optional) environments, as well as your controllers, terminals, and keyboards, are set up to display the national characters of the NLF you are installing.

The Japanese, and Korean NLFs use DBCS characters; they require the hardware and program products shown in Chapter 13, "Planning for Installation", on page 149.

The installation steps

Note: You must first install the QMF Version 7.2 base product before you can install a QMF National Language Feature. The QMF Version 7.2 distribution and production disks are required for the NLF installation.

Figure 33 on page 164 is an overview of the installation process.

Preliminary: Read the NLF program directory and complete the worksheet

The QMF NLF program directory contains information concerning the material and procedures associated with the installation of QMF. Because the program directory is updated between releases of QMF, it may contain useful information, including a description of PTFs and APARs, as well as modifications to this book. The program directory contains all the steps for installing QMF NLF from tape to disk and building the DCSS. Only the QMF NLF database installation into DB2 for VM is described in this book. You must complete the steps in the program directory before doing the installation steps in this book.

The following table shows the information that you need for NLF installation. Use it as your worksheet.

Table 31. Information Required during QMF NLF Installation (QMF V7.2 Worksheet)

Information required for:	**Your data:**	**QMF in DB2 workstation server**	**QMF in iSeries server**
Database/location	_____	_____	_____
Database type (DB2 VM, DB2 workstation server, or DB2 iSeries server	DB2VM	DB2WS	DB2400

Table 31. Information Required during QMF NLF Installation (QMF V7.2 Worksheet) (continued)

Prior QMF NLF level (if any)	_____	N/A	N/A
Q CONNECT password	_____	N/A	N/A
Default DBSPACE name for SAVE DATA command (default is DSQTSDEF)	_____	N/A	N/A

Step 1—Create the QMF NLF installation control file: DSQ2nCTL

The QMF exec, DSQ2*n*CTL, prompts you for information required during the NLF installation.

To create the QMF NLF installation control file, perform the following steps:

1. Access the QMF NLF distribution disk in WRITE mode
2. Fill in the worksheet shown in Table 31 on page 186, if you have not already done so.
3. Run the exec: DSQ2*n*CTL.

Prompts

You receive a series of prompts that ask you to supply the information you developed using the worksheet. The prompts vary, depending on the previous level of QMF, if any, installed on your system. (See "Step 1—Create QMF installation control file: DSQ2ECTL" on page 167.)

Anytime during this process, you can enter:

- `HELP` on the command line to receive more information.
- `CANCEL` to terminate the process before completion.

A file named QMFV720n INSTALL is created on your QMF NLF distribution minidisk. This file contains the information you supplied to the previous prompts.

If an installation file already exists from a previous installation, the information you enter is appended to this file and the previous information is deactivated.

Step 2—Run QMF NLF installation exec: DSQ2nINS

Before running this exec:

1. You must have access to the QMF NLF distribution disk in WRITE mode.
2. The QMF NLF installation control file QMFV720n INSTALL must exist on the QMF NLF distribution disk.
3. You must be linked to the DB2 for VM production disk in READ mode.

4. The following CMS commands allow the printer and console to continue processing unless a severe error is found.

```
spool prt cont hold
spool console start cont
```

Assumptions for installing QMF into a workstation database server

Before you attempt to install QMF on a remote database server, be sure to complete the prerequisites indicated in "installing QMF into a workstation database server on VM" on page 161.

Running the exec

To start the NLF installation exec, issue the command:

```
DSQ2nINS
```

The QMF NLF installation exec obtains its input from the QMF NLF installation control file. (See "Step 1—Create the QMF NLF installation control file: DSQ2nCTL" on page 187.) The QMF NLF installation exec performs the following steps:

1. Updates the Q.PROFILES and creates an NLF command synonyms table called Q.COMMAND_SYNONYM_*n*, if you are not migrating from any previous release of QMF.
2. Discards existing QMF NLF sample tables and creates new ones, if required.

Restart procedure

If this exec fails, use the following procedure to restart the exec and continue where you left off:

1. Determine what the problem is and fix it.
2. Rerun this exec with an input parameter equal to the restart value provided in the message when the exec terminated.

For example, if you get the message:

```
TERMINATING EXECUTION ...
TO RESTART THIS EXEC AND CONTINUE WHERE YOU LEFT OFF,
 - FIX THE PROBLEM ENCOUNTERED
 - RERUN THIS EXEC WITH THE INPUT PARAMETER OF 2
```

you can restart the exec with:

```
DSQ2nINS 2
```

Installation exec error messages

If you encounter a problem running the QMF installation exec, you need to find the error message describing the problem. Because this error message may be sent to either the console or the printer, you may want to spool your console and your printer to "HOLD".

Note: If you choose to spool your printer or console, enter the following to release the file that contains the error information:

```
spool prt close
spool console close
```

To transfer the printer files to your reader, issue the command:

```
TRANS PRT ALL *
```

Step 3—Start QMF NLF: DSQ2nINV

Follow "Step 4—Start QMF: DSQ2EINV" on page 171, noting the differences listed here. Remember that you can either tailor the QMF invocation exec (Step 4A) or invoke QMF from ISPF (Step 4B).

Step 3A—Tailor the QMF invocation exec: DSQ2nINV

Follow "Step 4A—Tailor the QMF invocation exec: DSQ2EINV (optional)" on page 171.

Modify the QMF NLF invocation exec, DSQ2nINV, to meet the requirements of your installation. The alterable parameters are the same as in Figure 36 on page 173. Note that DSQ2nINV is only a sample NLF invocation exec. The necessary links to disks, FILEDEFs, SQLINIT, and the ISPSTART commands are described clearly in simpler QMF invocation execs. These execs, DSQ2nIN1 (with ISPF) and DSQ2nIN2 (without ISPF), are located on the production disk. You may find them useful in constructing your own QMF invocation exec to match your environment requirements.

Step 3B—Invoking QMF from an ISPF environment (optional)

Follow "Step 4B—Invoke QMF from an ISPF environment (optional)" on page 176 and make changes to the ISPF Master Application Menu as shown in Figure 38 on page 190.

```
%--------------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(ISP@PRIM) NEWAPPL'
                2.'CMD(DSQ2EINV)'
                3.'CMD(DSQ2nINV)'
        /*                              */
        /* ADD OTHER APPLICATIONS HERE    */
        /*                              */
           P,'PANEL(ISPOPT)'
           X,'EXIT'
           ' ','.' '
           *,'?' )
)END
```

Figure 38. ISPF master application menu for NLF

Step 4—Run the IVP for QMF NLF interactive mode: DSQ2nIVP

Note: Be sure that QMF 7.2 is installed into the database you are using.

See "Step 5—Running IVP for QMF interactive mode : DSQ2EIVP" on page 177 for information on what the IVP does. Start QMF using the NLF program, DSQQMFn.

When you have successfully initialized QMF NLF, test for the help panel. To do this, press the "Help" key from the home panel. After you are on the help panel, press the "Cancel" key to take you back to the home panel.

You should then issue the command:

```
CONNECT Q (PASSWORD=xxx
```

where *xxx* is the value given to the CONNECT password of Q.

If you are running these jobs to any database other than DB2 for VM, the connect ID at that server must have DBA or SYSADM authority.

Test the QMF command interface

Test the command interface by issuing the following command:

```
CMS DSQ2nCI1
```

If this exec runs successfully, your QMF NLF profile is displayed, and you receive a message indicating that your CMS command was successful.

Test the QMF procedure

Run the IVP by issuing the following commands:

```
IMPORT PROC FROM DSQ2nIVP PROC *
RUN PROC
```

Answer YES to all prompts.

- If the procedure runs successfully, you receive a message indicating this.
- If the procedure does not run successfully, determine what the problem is by using the QMF NLF messages and message help panels.

Step 5—Install QMF NLF sample objects and application objects: DSQ2nSQD and DSQ2nSQI

After the QMF NLF is installed and verified, you can use the NLF to import the sample queries and procedures for the NLF.

If you have any previous version of this QMF NLF installed, you must delete the previous sample queries and procedures before installing QMF NLF V7.2 queries and procedures.

Perform the following steps to install the sample queries and procedures:

1. Start QMF if not already logged on.
2. Issue the command (if not done earlier):

   ```
   CONNECT Q (PASSWORD=xxx
   ```

   where “xxx” is the QMF CONNECT password of “Q”.

   If you are running these jobs to any database other than DB2 for VM, the connect ID at that server must have DBA or SYSADM authority.
3. Delete previous sample queries and procedures. (Run this step only if you have a previous version of this QMF NLF installed.)

   Import and run the procedure DSQ2nSQD as follows:

```
IMPORT PROC FROM DSQ2nSQD PROC *
RUN PROC
```

4. Install NLF sample queries and procedures, by importing and running procedure DSQ2nSQI with the following commands:

```
IMPORT PROC FROM DSQ2nSQI PROC *
RUN PROC
```

This procedure also installs the batch mode IVP and sample application procedures.

Restarting the procedure

If a failure occurs during this procedure, you can correct the error and run procedure DSQ2nSQD, which deletes any previously created sample queries. Then import and rerun procedure DSQ2nSQI to install sample queries and procedures.

Step 6—Run the IVP for QMF NLF batch mode (optional): DSQ2nBAT

Follow the directions for "Step 7—Run the batch-mode IVP (optional): DSQ2EBAT" on page 180.

To run the IVP, use the QMF invocation exec, specifying the parameters for batch mode and the QMF procedure Q.DSQ2nBAT, by issuing either of the following:

```
DSQ2nINV MODE(B) PROC(Q.DSQ2nBAT)
   or
DSQ2nINV MODE(B) PROC(Q.DSQ2nBAT) CMSSUB(NO)
```

Step 7—Post-installation cleanup

The QMF installation control file, QMFV720n INSTALL, resides on the QMF NLF production disk and contains the DB2 for VM CONNECT password for Q. This file was created in "Step 1—Create the QMF NLF installation control file: DSQ2nCTL" on page 187. Because this file is a potential security exposure, you should edit the installation control file and blank out the password. You may wish to change the DB2 for VM CONNECT password for Q and/or REVOKE DBA authority from Q, especially if you have chosen a non-trivial password for Q during QMF installation.

QMF uses the PROTOCOL (AUTO) option to run SQLINIT exec during Step 5. If the PROTOCOL (AUTO) option is not used at your machine, run SQLINIT to change the default PROTOCOL.

On the CMS command line, enter:

```
SQLINIT PROTOCOL(protocol)
```

where *protocol* is SQLDS, AUTO, or DRDA.

Part 3. Installing QMF on VSE/ESA

Chapter 16. Before You Begin

This chapter will help you plan for QMF installation. The key to success is having adequate resources. The following sections describe the hardware and software requirements, your planning considerations, and an overview of the installation task for QMF in the VSE/ESA environment.

Hardware

The required hardware consists of the following components:

Processor: You can install QMF for VSE/ESA on any processor supported by VSE/ESA Version 2.2 or later in ESA mode.

Tape Drive: You need a tape drive for loading the installation tape. You can use any tape drive supported by VSE/ESA Version 2.2 or later.

System console: you can use any terminal supported by VSE/ESA Version 2.2 or later.

Terminal: You need a terminal to install and test QMF. If you are installing support for a national language that requires the double-byte character set (DBCS), you will need a terminal that also supports DBCS to run the installation verification procedure (IVP).

Prerequisite software

The following table lists the program products with the minimum release levels required to support QMF for VSE/ESA Version 7.2. Later releases that are not available at the QMF Version 7.2 announcement time are not supported unless specifically stated otherwise.

Table 32. Prerequisite software for QMF for VSE/ESA Version 7.2

Required product	Version and release	Number
IBM Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA)	Version 2.2	5690-VSE
CICS/VSE	Version 2.2	5686-026
GDDM/VSE	Version 3.1	5686-057
DB2 for VSE	Version 7.1	5697-F42

The following table lists the program products with the minimum release levels required to support optional functions for QMF for VSE/ESA Version 7.2. Later releases that are not available at the QMF Version 7.2 announcement time are not supported unless specifically stated otherwise.

Table 33. Prerequisite software for optional functions for QMF for VSE/ESA Version 7.2

Product	Version and Release	Number
CHARTS (Interactive Chart Utility):		
GDDM-PGF (for GDDM Version 3.1.1)	Version 2.1.2	5668-812
GDDM-PGF (for GDDM Version 2.3)	Version 2.1.1	5668-812
Callable Interface Programs using the callable interface can be written in:		
IBM C/370 Compiler	Version 2	5668-187
C/370 Library	Version 2	5668-188
IBM HLASM	Version 1.1 or Version 1. 2	5696-234
VS COBOL II Compiler and Library	Version 1.4	5688-023
COBOL for VSE/ESA	Version 1.1	5686-068
PL/1 for VSE/ESA	Version 1.1	5686-069
User Edit Routines can be written in:		
IBM HLASM	Version 1	5696-234
VS COBOL II Compiler and Library	Version 1.4	5688-023
PL/1 for VSE/ESA	Version 1.1	5686-069
Governor Exit Routine:		
IBM HLASM	Version 1	5696-234

QMF storage requirements

Before you start using QMF, you need to make sure that each CICS partition that runs QMF has enough storage to accommodate QMF programs and the QMF reports users create.

The partition must be large enough to accommodate:

- **All QMF phases**: 2.8 MB 31-bit storage, total
- **Storage for users to execute queries and hold QMF report data**: Average of 0.5 MB to 1 MB GETVIS storage per user. Some report options may require additional storage.

You can allocate storage for both purposes above 16 MB.

For a QMF system with up to 20 users, allocate at least 24 MB virtual storage for your CICS partition. The minimum acceptable partition size for any QMF system is 18 MB, regardless of the number of users.

To specify your partition size, use the VSE ALLOC statement in the ALLOC.PROC data set of the IPL procedures, as shown in the following example. For systems with more than 20 QMF users, increase the ALLOC 0.5 MB to 1 MB for each additional user.

Also allow 9 MB, within the 24 MB, for your programs. Specify this space with the SIZE value in the IPL allocation data set:

```
// JOB ALLOC
// EXEC LIBR,PARM='MSHP'
ACC S=IJSYSRS.SYSLIB
CATALOG ALLOC.PROC    DATA=YES    RELPLACE=YES
ALLOC S,F1=24M
SIZE F1=9m
⋮
```

Some users might require more than 1 MB of GETVIS storage if they use complex formatting options for a report, or if a large amount of data is returned from a query. "Adjusting GEVIS Storage Used for Report Data (DSQSBSTG) on page xx explains how to calculate the desired allocation and size.

If a QMF transaction runs out of storage in the CICS partition, it might time out waiting for storage to become available. THerefore, when you adjust the GETVIS values for each user, make sure you increase the ALLOC to accommodate that additional storage.

You might also consider allocating a spill file for certain users, by defining CICS auxiliary temporary storage in DFHTEMP. The spill file is used for extra storage for data and reports.

The installation procedure in this book installs QMF to an individual CICS partition. If you have several users using QMF in different CCS partitions, you might consider lading QMF into the 31-bit shared virtual area. Or, if the QMF users you support routinely use more than 1 MB of GETVIS storage for queries and reports, you might also consider using CICS multiregion operators or intersystem communications to provide more efficient use of CICS resources at your site.

Apply service

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 program temporary fixes (PTFs) for QMF and its prerequisite products. Additionally, request QMF's preventive service planning (PSP) bucket, SUBSET: QMFVSE under UPGRADE QMF720. the bucket contains general hints, HIPER APARs and documentation changes. Subscribers that have access to either Information/Access or ServiceLink, can download this information directly.

Check space requirements

To ensure that there is adequate disk storage for QMF installation, you need to account for three kinds of space requirements. You must calculate all three requirements to get an accurate storage estimate.

Library space

Your first task is to calculate your exact library space requirements in blocks. You perform this calculation as part of the MQF installation described in the QMF Program Directory. The number of QMF library blocks can vary. Although there is a set number of blocks for the QMF base product, the library block number increases if you are adding support for a national language in addition to English.

If you need a rough estimate for planning purposes only, the number of library blocks is approximately 14,100 and the minimum number for each national language feature (NLF) is approximately 4,100.

VSAM catalog

In addition to the library retirement, QMF needs to define a file in the VSAM space. This file needs:

- 2.5 MB of free VSAM space in a VSAM catalog.
- 0.5 MB of free VSAM space in the catalog that holds the GDDM file, ADMF. You need free storage in the user catalog where the ADMF file resides.

If you are using an NLF: For each NLF, you will need an additional 2.5 MB of VSAM catalog space and a corresponding 0.5 MB space in the ADMF file.

dbspace

When you installed DB2, you created public and private dbspaces. QMF needs some of the public dbspaces for tables, queries, procedures, forms, and data.

A dbaspace is a logical allocation of space in the database, which consists of 4K pages. To convert dbspaces to megabytes, cylinders and tracks, or to add dbspaces to the database, see the *DB2 Server for VSE System Administration* manual.

If QMF tries to acquire the dbspace and there is not an exact match, it will try to acquire the next largest size available. To avoid wasting space, check that you have:

- One 5,120-page public dbspace
- Three 256-page public dbspaces
- Six 128-page public dbspaces

QMF needs this amount of space for each DB2 database; if you have multiple databases, you need to take that into account. To verify the seize of the dbspaces, you can enter the following SQL statement from ISQL:

```
SELECT * FROM SYSTTEM.SYSDBSPACES WHERE DBSPACETYPE=1 AND OWNER=' '
```

Your disk storage allotment is the sum of the dbspace, VSAM catalog space and library block size calculated in cylinders or blocks.

Check your CICS partition size

The minimum acceptable partition size for any QMF system is 18 MB, regardless of the number of users. For a QMF system with up to 20 users, allocate 24 MB virtual storage for your CICS partition. To specify your partition size, use the VSE ALLOC statement in the IPL procedure ALLOC.PROC, such as:

```
ALLOC F4=24M
SIZE=F4=9M
```

For systems with more than 20 QMF users, increase the ALLOC by 0.5 MB to 1MB for each additional user. Also allow 9 MB , within the recommended 24 MB, for your programs. You specify this space with the SIZE value in the IPL allocation data set.

Because of the size of the GETVIS area is the difference between the partition size and the SIZE value, your GETVIS is 15 MB. After installation, you can adjust GETVIS space to maximize storage for a user's queries and reports.

Partition size for installation

You need a partition to run the QMF installation job. His partition must have a partition size of at least 1.5 MB.

The planning considerations

Not all QMF installations are the same. The following sections describe some additional installation considerations that might apply to your situation.

Tailoring GDDM for QMF and CICS

Before you install QMF, GDDM must be fully installed, tailored, and tested. It is important to do a complete GDDM installation and not merely restore to a library. During QMF installation, QMF modifies GDDM's ADDMF file. Additionally, you must define GDDM resources, such as programs and transactions, to CICS.

Changing GDDM Version 2.3 default parameters

If you are using GDDM Version 2.3, you might need to modify a parameter in the GDDM external defaults module. Ensure that the IOSYNCH parameter in ADMADFC is set to YES.

Run the installation verification procedure (IVP) for GDDM

Check that you have GDDM properly installed by running the IVP for GDDM. The IVP minimizes installation problems and ensures that you are installing QMF onto a clean system.

Running DB2 guest sharing

You have the option to connect your CICS partition to a DB2 database on either VSE or VM. When VSE is a guest to VM and shares data with the host's applications through a common VM DB2 database, it is called *guest sharing*. The benefit of SQL guest sharing is that both VM and VSE users can use a common database. QMF requires minimum levels of VM/ESA Version 1.1 and DB2 Version 6 to use QMF in an SQL guest sharing environment. You do not have to install QMF for VM.

If you establish an SQL guest sharing environment and want to install QMF on VSE, complete the installation as if VSE owned the database. The VM DB2 database is transparent to the VSE user.

However, if you have both DB2 and QMF installed under VM/ESA, and you want to install an additional QMF product in VSE, you can skip the part of the QMF installation that deals with database installation. During the install process, you are told when to skip that step. Otherwise, the remainder of this manual assumes that you are installing QMF into a DB2 for VSE database.

Customizing DB2 for double-byte character support

If you plan to install QMF with a national language that requires double-byte character support, you need to complete the database customization before installing QMF.

Installation overview

Every data center is different, and every system in each data center can be configured numerous ways. You might have multiple DB2 databases, on one or more CICS systems, VSE running as a guest under VM/ESA, DB2 guest sharing, another version of QMF (either on VM or VSE), or special national

language requirements. Because your VSE system might be unique, the QMF installation has been designed so that you can easily install and re-install using only a few simple jobs.

Some of these jobs will be run only once, others might be run multiple times depending on your configuration.

Base installation

The base (English version) installation requires that you run the following jobs:

DSQ3INIT

This job establishes the initialization criteria for the remainder of the installation. The other installation jobs use the procedure cataloged by DSQ3INIT to find information about installed products such as GDDM, DB2, and VSAM. It also contains information about the QMF installation. You run DSQ3INIT only once.

DSQ3EINS

This job defines and loads the QMF panel file (DSQPNLE) into VSAM space. The panel file contains all of the panel definitions. DSQ3EINS also loads maps and sample charts; you run it only once.

DSQ3EDBI

This job is QMF's database installation job. DSQ3EDBI creates QMF control tables, loads QMF packages and defines and loads sample tables into the DB2 database. You run this job once for each local SQL database that you are connecting to CICS.

Normally, running this job is mandatory. However, if you currently have QMF for VM/ESA Version 7.2 or later installed, and you want to add a QMF for VSE/ESA in an SQL guest sharing environment, you do not need to run this job. This is because the database portion of QMF was installed during the QMF VM installation. However, if you want to define a DB2 VSE database in addition to the DB2 VM database, you do need to run DSQ3EDBI.

DSQ3ELNK (optional job) Here

This job link-edits QMF with the current versions of GDDM and DB2. You do not need to run this job if you have installed:

- GDDM/VSE Version 3.2
- CICS for VSE/ESA Version 2.3
- DB2 for VSE Version 7.1

QMF is already linked with those versions.

Installing language support

The QMF base installation loads all of the panels and maps in English. If you require QMF in another or different language, you need to order one or more of the National Language Features (NLFs) of QMF.

You can distinguish base installation members from NLF members by a single character abbreviation known as the National Language Identifier (NLID). In the manual, and throughout the QMF library, we use the letter *n* to represent the NLID. For example, if you are installing DSQ3FINS.Z for French support, you substitute *F* in place of the *n* in member name DSQ3*n*INS.Z. The same member name for English support, DSQ3EINS.Z., lists all of the languages and their associated NLIDs.

Table 34. NLIDs representing QMF base (English) and National Language Features (NLFs)

Language	**NLID (n)**
Brazilian Portuguese	P
Canadian-French	C
English	E
French	F
German	D
Italian	I
Japanese	K
Korean	H
Spanish	S
Swiss French	Y
Swiss German	Z
Uppercase English	U

The uppercase feature (UCF) uses the English language, but converts all test 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.

NLF install process

You begin an NLF installation by scanning the tape for the base product. After the tape is restored to disk, you continue through the installation procedures for the base install, up to the point of CICS tailoring. You must complete the

installation of the base product, up to CICS tailoring, before you install the NLFs. While doing the base installation, ignore the sections in the procedures that apply to NLFs.

After the base product is installed, go back through the same procedures and follow the directions that apply for an NLF. Continue with CICS tailoring and customize both the base and the NLFs at the same time. Last, run the Installation Verification Procedures (IVPs) for the base product and for each NLF.

The NLF-specific installation jobs are:

- **DSQ3nINS** This job is the same as DSQ3EINS, except that it loads the panel files in your national language. This is a mandatory step.
- **DSQ3nDBI** This job is the same as DSQ3EDBI, except that it loads the sample tables and profile table in your national language. This is a mandatory step for each database that you connect to CICS.
- **DSQ3nLNK** This job is the same as DSQ3ELNK, except it link-edits the national language parts of QMF. This is an optional step depending on the product versions you have installed.

CICS tailoring

CICS can be tailored after the product base and NLFs have been installed. CICS can be tailored for both the base and any NLFs at the same time. However, modifications must be made to every CICS system that works with QMF.

During CICS tailoring, four CICS tables must be modified:

- Destination control table (DCT)
- File control table (FCT)
- PROGRAM resource (CSD) or processing program table (PPT)
- TRANSACTION resource (CSD) or program control table (PCT)

All four tables can be modified by assembling resource definition tables. Some of the tables can also be modified by defining resources online (RDO) in a CICS system definition (CS) data set. Use copyboooks and copy statements to change the tables. Similar copybooks and copy statements should exist for the NLF versions.

When CICS tailoring has been completed, the installation must be checked by following the IVPs for the base product and for each NLF.

Chapter 17. Tailoring Your Installation

QMF must be customized before it can be used. This chapter lists the necessary steps to tailor or customize QMF and CICS for your system. The QMF for VSE Version 7.2 installation must be completed in order to proceed with this chapter.

Punch members to an editor

Members must be punched to a facility that have an editor such as ICCF or Virtual Machine (VM) because they cannot be edited in a VSE sublibrary. The following procedure illustrates how to punch QMF jobs to an ICCF library:

1. Return to the main VSE/ESA Function Selection panel.
2. Select the Command Mode option to enter commands directly. You can switch to a secondary library by entering

   ```
   /SWI nn
   ```

   where nn represents the target ICCF library number.
3. Punch the following members to ICCF. Press Enter after typing each command.

   ```
   LIBRP PRD2.PROD DSQ3INIT.Z DSQ3INIT (REPLACE
   LIBRP PRD2.PROD DSQ3EINS.Z DSQ3EINS (REPLACE
   LIBRP PRD2.PROD DSQ3EDBI.Z DSQ3EDBI (REPLACE
   ```
4. For NLF: Punch NLF installation members to ICCF using the NLID for your NLF listed in Table 34 on page 202. Press Enter before typing each command.

   ```
   LIBRP PRD2.PROD DSQ3nINS.Z DSQ3nINS (REPLACE
   LIBRP PRD2.PROD DSQ3nDBI.Z DSQ3nDBI (REPLACE
   ```

install QMF base

This section covers installing the QMF base, which involves:

- Modifying and cataloging the initialization procedure
- Running the install job
- Installing to the DB2 database

Catalog the initialization procedure

DSQ3INIT is the QMF initialization procedure. It establishes installation criteria for the remainder of the installation, which is stored in DSQ3INIT.PROC. Because the information stored in DSQ3INIT is critical to

the success of the installation, ensure your entries are correct before running the job. An incorrect entry in this job will cause errors in subsequent job steps.

You must edit DSQ3INIT before it can run:

1. Delete the first line of the file, which begins with CATALOG.
2. Change all instances of ..* to * with the following command:

   ```
   ch /..*/*/ *
   ```

3. Delete the two last lines of the file, leaving the end-of-job statement.
4. Verify or change the name of the QMF library and sublibray, if necessary. If you are using anything other than the default library, PRD2.PROD, change the name to your library name:

   ```
   // EXEC LIBR,PARM='MSHP'
      ACC S=PRD2.PROD
      CATALOG DSQ3INIT.PROC

   // SETPARM QMFLIB=PRD2          *-- QMF for VSE LIBRARY
   // SETPARM QMFSLIB=PROD         *-- QMF for VSE SUBLIBRARY
   ```

5. Check the default library and sublibray for GDDM/VSE and DB2 for VSE. Compare, and change if necessary, the defaults to actual library and sublibrary names being used. The default values are:

   ```
   // SETPARM ADMLIB=PRD2        *-- GDDM/VSE LIBRARY
   // SETPARM ADMSLIB=PROD       *-- GDDM/VSE SUBLIBRARY
   // SETPARM SQLLIB=PRD2        *-- DB2 LIBRARY
   // SETPARM SQLSLIB=DB2720     *-- DB2 SUBLIBRARY
   ```

6. Check the VSAM catalog name and ID. The catalog name and ID specify the target VSAM catalog for the QMF panel file and any NLF panel files. Compare the fields and change, if necessary:

   ```
   // SETPARM UCAT=VSESPUC                *-- FILE NAME OF CATALOG
   // SETPARM UCATID='VSEP.USER.CATALOG'

   *-- FILE ID OF CATALOG
   ```

7. Determine whether you changed any of the defaults for GDDM/VSE. ADMF should have been defined in a VSAM catalog during the GDDM install. The QMF installation loads maps and forms to ADMF. QMF requires the file ID of ADMF (ADMFID), the catalog name (ACAT), and the catalog ID (ACATID).

   If you changed the defaults, change the following statements to match your naming convention:

   ```
   // SETPARM ADMID='ADMF'            *-- FILE ID OF GDDM/VSE FILE ADMF
   // SETPARM ACAT=VSESPUC            *-- CATALOG NAME AND CATALOG ID IN
   // SETPARM ACATID='VSESP.USER.CATALOG'     *-- WHICH ADMF IS DEFINED
   ```

8. File the job and run DSQ3INIT. Check the system console to ensure that the job ran with a return code of 0.

   If the job did not run with a return code of 0:

a. Check for an error message on the system console.
b. Check the list output to find the cause of the problem.
c. Correct the problem.
d. Return DSQ3INIT.
e. Recheck the return code.

Run the QMF installation job

DSQ3EINS is the QMF installation job:

- It defines and loads the QMF panel file
- It loads sample charts
- It loads maps

To successfully load and execute this job, these tasks must be accomplished during the initial installation:

1. Edit DSQ3EINS to change or supply the required parameters
   a. Delete the first line of the file that starts with CATALOG.
   b. Change all instances of ..* to * with the following command:

      ```
      ch /..*/*/ *
      ```

   c. Delete the two last lines of the file, leaving the end-of-job statement.
2. Return to the top of the job and ensure the four job steps are set to YES. DSQ3EINS contains four job steps When a SETPARM is set to YES, it indicates that the step will be run; when set to NO, the step is skipped. Under most circumstances, these steps are run only once because VSE can share files with multiple CICS systems. For subsequent installations, or under error conditions, some of the job steps might need to be set to NO.

   ```
   // SETPARM STEP1=YES      *-- DEFINE CLUSTER DSQPNLE
   // SETPARM STEP2=YES      *-- LOAD DSQPNLE
   // SETPARM STEP3=YES      *-- LOAD QMF CHARTS
   // SETPARM STEP4=YES      *-- LOAD QMF MAPS
   ```

3. Verify or change all instances of the library and sublibray names. If you installed QMF to a library other than PRD2.PROD, you must change the library and sublibrary names.
4. Supply one or more volume IDs for the VSAM cluster that holds the QMF panel file. Locate this cluster definition under job step 1.

   ```
   // EXEC IDCAMS,SIZE=AUTO
    DEFINE CLUSTER ( -
           NAME (QMF720.DSQPNLE ) -
           RECORDS (1200 50) -
           SHAREOPTIONS (3) -
           RECORDSIZE (1920  32756 ) -
           VOLUMES (--V001--) -
   ```

   Replace the variable -V001- with the volume IDs , such as DOSRES or SYSWK1. For example:

```
VOLUMES (DOSRES SYSWK1) -
```

or

```
VOLUMES (DOSRES) -
```

5. File and run the job. Check the system console to ensure that the job ran with a return code of 0.

Install QMF base into DB2 database

Run DSQ3EDBI, the database installation job, for each SQL database that you are connecting to QMF.

Skip this procedure if you have a DB2 guest sharing environment and have installed QMF for VM. Under these conditions, you can continue with "Tailor QMF for NLF" on page 209, if you have an NLF to install, or skip to "Link-edit jobs for QMF" on page 211 for a base installation.

DSQ3EDBI

- Creates QMF control tables
- Loads QMF packages
- Defines and loads sample tables

1. Edit DSQ3EDBI (first use of the job only).
   a. Delete the first line of the file, starting with CATALOG.
   b. Change all instances of ..* to * with the following command:
      ```
      ch /..*/*/ *
      ```
   c. Delete the last two lines of the file, leaving the end-of-job statement.
2. Ensure that the first three job steps are set to YES.
   ```
   // SETPARM STEP1=YES    -- CREATE QMF CONTROL TABLES IN SQL DB
   // SETPARM STEP2=YES    -- LOAD QMF PACKAGES INTO SQL DB
   // SETPARM STEP3=YES    -- LOAD QMF SAMPLES INTO SQL DB
   ```

   For subsequent installations, or under error conditions, some of the job steps might need to be set to NO. Changing the setparm to NO will skip the step.
3. Locate the
   ```
   // SETPARM DBNAME=SQLDS
   ```

   parameter and verify or change SQLDS to the name of the database that you are using. DBNAME is the database name that is specified in the SQL DBNAME directory.
4. Determine whether this database has an earlier version of QMF for VSE installed. Possible earlier versions are Version 3.3, Version 6.1, or Version 7.1.

If an earlier version is installed, located the following statement and change the value from NO to Version 3.3, Version 6.1, or Version 7.1:

```
// SETPARM MIGRATE=NO
//
```

Using this parameter will prevent duplicate control tables from being created.

5. Verify or change all instances of the QMF library and sublibray names. If QMF was installed to a library other than PRD2.PROD, change the library and sublibrary names.
6. Ensure that DB2 is up and running in multiple user mode.
7. Ensure the SQLDBA's password is set to SQLDBAPW.

   The QMF installation procedures assume that the password for SQLDBA is set to SQLDBAPW. IF the password is set to anything else, DSQ3SETQ.A must be updated and re-cataloged into the QMF installation library.

   To update DSQ3SETQ.A:

   a. Punch the member DSQ3SETQ.A to an editor, such as ICCF.
   b. Modify the CONNECT statement by replacing SQLDBAPW with the existing password for SQLDBA:

      ```
      CONNECT SQLDBA IDENTIFIED BY 'new-SQLDBAPW'
      ```

   c. Recatalog DSQ3SETQ.A into the QMF installation library (PRD2.PROD).
8. File and run DSQ3EDBI. As the job runs, the system console displays messages that indicate which job step is executing. At the end of the job, check the system console to ensure that job completed with a return code of 0. Those who are migrating from an earlier version of QMF for VSE, might get a return code of 6 in job step; in this case, the return code can be ignored.

   If the job did not complete with a return code of 0 or 6, follow the same tasks as in step 5 on page 208, except Rerun the job DSQ3EDBI instead of DSQ3EINS for task f.

   To install QMF into additional DB2 databases, repeat the procedure for each database, starting with step 3 on page 208.

Tailor QMF for NLF

Two members, DSQ3*n*INS and DSQ3*n*DBL, should have been punched to add support for a national language. These jobs need to be edited and executed.

Install NLF

Like DSQ3EINS, DSQ3*n*INS is run only once.

1. Edit DSQ3nINS for the following:
   a. Delete the first line of the file, starting with CATALOG.

b. Change all instances of ..* to * with the following command:

```
ch /..*/*/ *
```

c. Delete the last two lines of the file, leaving the end-of-job statement.

d. Locate the three setparms and ensure that they are set to YES:

```
// SETPARM STEP1=YES    *-- DEFINE CLUSTER DSQPNLn
// SETPARM STEP2=YES    *-- LOAD DSQPNLn
// SETPARM STEP3=YES    *-- LOAD QMF MAPS TO ADMF
```

e. Verify or correct the QMF library and sublibrary names. Change the names accordingly if they were changed from the default PRD2.PROD library.

f. Supply one or more volume IDs for the VSAM cluster that holds the national language version of the QMF panel file. Locate this cluster definition under job step 1:

```
// EXEC IDCAMS,SIZE=AUTO
 DEFINE CLUSTER ( -
   NAME (QMF720.DSQPNLn ) -
   RECORDS (1200 50) -
   SHAREOPTIONS (3) -
   RECORDSIZE (1920  32756) -
   VOLUMES (--V001--) -
```

2. File DSQ3*n*INS and run the job.

Install QMF into SQL database

Edit DSQ3nDBI for each database that needs NLF support:

1. Delete the first line of the file, starting with CATALOG.
2. Change all instances of ..* to * with the following command:

   ```
   ch /..*/*/ *
   ```

3. Delete the last two lines of the file, leaving the end-of-job statement.
4. Ensure that the first three job steps are set to YES:

   ```
   // SETPARM STEP1=YES   *-- Create profile
   // SETPARM STEP2=YES   *-- Drop sample tables
   // SETPARM STEP3=YES   *-- Create and load sample tables
   ```

   For subsequent installations, or under error conditions, some of the job steps might need to be set to NO. Changing a setparm to NO skips the step.
5. Locate the database name SQLDS, and verify that it is set for the correct database:

   ```
   // SETPARM DBNAME=SQLDS    *-- TARGET DB2 DBNAME FOR QMF on VSE/ESA
   ```

6. Determine whether this database has an earlier version of QMF for VSE installed.

   If an earlier version is installed, locate the following statement and change the value from NO to that version, for example V6R1:

```
// SETPARM MIGRATE=NO
//
```

7. Verify or correct the QMF library and sublibrary names. Change the names accordingly if they were changed from the default PRD2.PROD library.
8. Ensure that DB2 is up and running in multiple user mode.
9. File DSQ3*n*DBI and run the job. Check the system console to ensure that the job completed with a return code of 0.

Link-edit jobs for QMF

QMF is prelinked with the following release levels of products:

- GDDM/VSE Version 3.2
- CICS/VSE Version 2.3
- DB2 Version 6

If you have different releases of these products, the following job must be run:

1. Punch DSQ3ELNK to a library and edit the job. The following example uses ICCF to perform the punch:

   ```
   LIBRP PRD2.PROD DSQ3ELNK.Z DSQ3ELNK (REPLACE
   ```

   Press Enter.
2. Delete the first line of the file, starting with CATALOG.
3. Change all instances of ..* to * with the following command:

   ```
   ch /..*/*/ *
   ```
4. Verify or correct the QMF library and sublibrary names. Change the names accordingly if they were changed from the default PRD2.PROD library.
5. Verify or change the search chain so that it contains the library and sublibrary for QMF, DB2, GDDM/VSE, and CICS:

   ```
   // LIBDEF OBJ,SEARCH=(PRD2.PROD,PRD2.DB2710,PRD1.BASE)
   ```
6. File and run the job; check the system console to ensure the job completed with a return code of 4. A return code of 0 is not returned because of weak external references (WXTRNs) that are not resolved during the linkage editor run.

   If the job did not complete with ar return code of 4, recheck the LIBDEF statement for the above products and rerun the link-edit job. Below is an example of the output from this link-edit:

   ```
   21651 WARNING - RMODE=ANY ASSIGNED TO PHASE, BUT THE PHASE
         CONTAINS 2 AND/OR 3 BYTE RELOCATABLE ADDRESS CONSTRAINTS
   UNRESOLVED EXTERNAL REFERENCES            WXTRN    ADMUFO
                                             WXTRN    GERHND
                                             WXTRN    ADME000C
                                             WXTRN    ADMADFC
                                             WXTRN    ADMACIN
   ```

```
WXTRN    ADMUOFF
WXTRN    DSQCLDQ
WXTRN    LTTBAS
WXTRN    LTTBASX
WXTRN    DSNHLI
WXTRN    DSQIRDB2
```

Additionally, several messages will appear about the use of WXTRNs and using 2- or 3-byte ADCONs. These messages are expected and should not cause a problem for QMF.

Link jobs for NLF

Repeat the procedure in "Link-edit jobs for QMF" on page 211 for your NLF version of DSQ3*n*LNK.

Tailor CICS

The following CICS tables must be modified for QMF to run in CICS:

- Destination control table (DCT)
- File control table (FCT)
- PROGRAM resource (CSD) or processing program table (PPT)
- TRANSACTION resource (CSD) or program control table (PCT)

These modifications must be made for every CICS system that works with QMF.

Resources, such as QMF programs and transactions that CICS controls for a particular run, must be defined. They can be defined online (RDO) in a CICS system definition (CSD) data set, or by assembling resource definition tables using macros.

The macro method must be used to define the FCT and DCT. QMF supplies copybooks for modifying these tables.

The PCT and PPT can also be modified with the QMF supplied copybooks. However, the recommended method is to modify these tables with an RDO to a CSD data set. RDO allows one to interactively create resource definitions and store them in a data set.

CICS offers a utility program (DFHCSDUP) to update the CSD with a batch job. QMF provides a job that defines the QMF programs and transactions to CICS without reassembling the DFHPCT and DFHPPT.

The following procedures require knowledge of locating tables and assembling them using the Interactive Interface. This is described in the *IBM VSE/ESA Administration* guide.

Modify the DFHFCT and DFHDCT

Change these tables using the macro method.

Before editing, locate the source used to create the FCT and DCT for this CICS system. In the following examples, it is assumed that the skeletons provided with VSE/ESA to bring up another CICS system were used. If these skeletons were not used, the CICS for VSE/ESA Resource Definition (Macro) can be used to locate the appropriate place to insert the changes. Note that these examples have a suffix of C2; your tables may be different.

Modify DFHFCT

Modify the source of your DFHFCT to define the QMF panel file to CICS:

1. Find the LIBDEF statement and ensure that the search chain contains the library and sublibrary for QMF. Otherwise, VSE/ESA will not be able to find the copybook.

   ```
   // LIBDEF *,SEARCH=(PRD1.BASE,PRD2.PROD)
   ```

2. Add a local entry for the QMF panel file in the FCT. If an NLF is to be added, add a copy statement for the NLF member:

   ```
   *-------------------------------------------------
   *      LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
   *-------------------------------------------------
          COPY DSQ3EFCT
          COPY DSQ3nFCT
          SPACE 3
   ```

   Substitute the appropriate NLID for the n.

3. Assemble and link-edit the member to create a new DFHFCTC2 phase.

Modify DFHDCT

Locate the source of your DFHDCTC2 and make the following changes:

1. Find the LIBDEF statement and ensure that the search chain contains the library and sublibrary for QMF. Otherwise, VSE/ESA will not be able to find the copybook.

   ```
   // LIBDEF *,SEARCH=(PRD1.BASE,PRD2.PROD)
   ```

2. Find the local entry for TYPE=SDSCI and add a copy statement for DSQ3DCTS as shown in the example below:

   ```
   *-------------------------------------------------------
   *      OTHER LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
   *-------------------------------------------------------
          COPY DSQ3DCTE
          SPACE 3
   ```

3. Assemble and link-edit the member to create a new DFHDCTC2 phase.

   Ensure the job completes with a return code of 0 or 4. If you receive higher return codes, check the list output and correct the error.

Define QMF programs and transactions to CICS

QMF provides help to define the QMF programs and transactions to CICS:

- By providing a batch job that defines the QMF resources to the CICS CSD
- By providing copybooks that can be included in the CICS PPT and PCT

Define the QMF programs to the CSD using the job DSQ3ECDS (and DSQ3*n*CSD for NLF installs).

Update the CSD

This procedure creates a new LIST called QMF, which is defined in the CSD. Additionally, for each language, a GROUP called QMF720*n* is defined in the LIST QMF. QMF720*n* contains the definition of the QMF programs and transactions (E for English).

1. Punch the following member to ICCF using this command:

   ```
   LIBRP PRD2.PROD DSQ3ECSD.Z DSQ3nCSD (REPLACE
   ```

   Press Enter.
2. Punch equivalent NLF members to ICCF. Substitute the NLID for the n in the following example:

   ```
   LIBRP PRD2.PROD DSQ3nCSD.Z DSQ3nCSD (REPLACE
   ```

   Press Enter.
3. Edit DSQ3ECSD (or DSQ3*n*CSD for NLF)
   a. Delete the first line of the file, starting with CATALOG.
   b. Change all instances of ..* to * with the following command:

      ```
      ch /..*/*/ *
      ```

   c. Delete the last two lines of the file, leaving the end-of-job statement.
4. Check that the file ID and catalog reflect your CSD:

   ```
   // DLBL DFHCSD,'CICS.CSD',,VSAM,CAT=VSESPUC
   ```
5. Verify that the library and sublibrary names in the POWER statements are the QMF library and sublibrary names:

   ```
   * $$ SLI MEM=DSQ3ECDN.A,S=PRD2.PROD
   * $$ SLI MEM=DSQ3BCDB.A,S=PRD2.PROD
   ```

   For NLF:

   ```
   * $$ SLI MEM=DSQ3nCDN.A,S=PRD2.PROD
   ```
6. File and run the job. Ensure the job completes with a return code of 0 or 4.

Run CEDA

CEDA is one of three interactive online transactions that comprise RDO. CEDA can be used to modify, delete, check and browse definitions while CICS is running. It provides commands for managing groups and lists, which include installing a group of resource definitions on an active system.

To activate resource definitions from the main VSE/ESA Function Selection panel:

1. Select 7, CICS - Supplied Transactions, and press Enter.
2. Select 2, Invoke CEDA , from the CICS-Supplied Transaction panel and press Enter.
3. Type

   ```
   AP LIST (QMF) TO(VSELIST)
   ```

   and press Enter.

   The VSELIST parameter must be the name specified for the GRPLIST parameter specified in DFHSIT of this CICS. CEDA appends the LIST QMF to the LIST VSELIST and ensures that the QMF definitions are known to CIC after the next cold start.

   To activate these QMF definitions immediately, perform a CICS cold start, or issue:

   ```
   CEDA INSTALL GR(QMF720e)
   ```

   for a temporary, but immediate, change.
4. Repeat the procedure for applicable NLF members:

   ```
   CEDA INSTALL GR(QMF720n)
   ```

Modify the DFHPCT and DFHPPT

If the QMF programs and transactions were not defined to the CSD, the following modifications must be made:

Modify the DFHPCT

1. Locate the source of your DFHPCT.
2. Find where local entries are made and enter copy statements for DSQ3EPCT and for any NLF PCT entries as shown in the example below:

   ```
   *-------------------------------------------------
   *      LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
   *-------------------------------------------------
          SPACE 3
          COPY DSQ3EPCT     ***** QMF for VSE BASE ENTRIES
   ```

   For NLF:

   ```
          COPY DSQ3nPCT     ***** QMF for VSE NLF ENTRIES
   ```

   Substitute the appropriate NLID for the n.
3. Assemble and link-edit the member to create a new DFHPCTC2 phase.

Modify the DFHPPT

1. Locate the source of your DFHPPT.
2. Find where local entries are made and enter copy statements for DSQ3EPCT and for any NLF PCT entries as shown in the example below:

```
*-------------------------------------------------------
*      LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
*-------------------------------------------------------
       SPACE 3
       COPY DSQ3EPPT     ***** QMF for VSE BASE ENTRIES
```

   For NLF:

```
       COPY DSQ3nPPT     ***** QMF for VSE NLF ENTRIES
```

   Substitute the appropriate NLID for the n.
3. Assemble and link-edit the member to create a new DFHPPTC2 phase.

Modify the CICS startup job

1. Locate the LIBDEF statement with the following search string and ensure that it contains the QMF library and sublibrary:

```
// LIBDEF *,SEARCH+(PRD2.CONFIG,PRD1.BASE,PRD2.DB2710, -
                PRD2.PROD),PERM
```

2. Define the labels for the base QMF panel file and for the NLF equivalent member with your other CICS DLBL statement:

```
// DLBL DSQPNLE,'QMF720.DSQPNLE',,VSAM,CAT=VSESPUC
```

   For NLF:

```
// DLBL DSQPNLn,'QMF720.DSQNLE',,VSAM,CAT=VSESPUC
```

   Optionally, the above DLBL statement can be included in the system standard label procedure.
3. Ensure that virtual storage allocated to the CICS partition was expanded.

Shut down and restart CICS to incorporate changes made to the CICS tables and to the CICS startup job

---

Install QMF for VSE/ESA into a second CICS system

When installing QMF to several CICS systems, determine if a single CSD is shared among the CICS systems or individual CSDs per CICS system. Determine if all the CICS systems use the same CSD by comparing the GRPLIST parameter in the SIT for each CICS.

If there is a single CSD for all your CICS systems, further customization steps may be skipped after QMF programs and transactions to the CSD have been initially defined. The FCT and DCT of the second CICS must still be modified.

If there are individual CSDs for every CICS system, the entire CICS customization procedure must be repeated for each CICS system.

Chapter 18. Installing QMF into Remote Database Servers

TCP/IP communications must be in place between the local DB2 for VSE requester and the remote database server in order to install QMF into remote database servers from VSE.

Installing QMF V7.2 into a DB2 Universal Database remote server

DB2 Universal Database Version 5 or higher is supported

Punch Members to an editor

Members must be punched to a facility that have an editor such as ICCF or VM because they cannot be edited in a VSE sublibrary. The following procedure illustrates how to punch QMF jobs to an ICCF library:

1. Return to the main VSE/ESA Function Selection panel.
2. Select the Command Mode option to enter commands directly. You can switch to a secondary library by entering

   ```
   /SWI nn
   ```

   where nn represents the target ICCF library number.
3. Punch the following members to ICCF. Press Enter after typing each command.

   ```
   LIBRP PRD2.PROD DSQ3EDBU.Z DSQ3EDBU (REPLACE
   LIBRP PRD2.PROD DSQ3BPKG.Z DSQ3BPKG (REPLACE
   ```
4. For NLF: Punch NLF installation members to ICCF. Using the NLID for your NLF listed in Table 34 on page 202. Press Enter before typing each command.

   ```
   LIBRP PRD2.PROD DSQ3nDBU.Z DSQ3nDBU (REPLACE
   ```

Installation steps

The steps listed here describe a QMF server installation to a remote DB2 UDB Version 5 or higher database server. The QMF server installation utilizes DB2 for VSE batch requester services and assumes that all connections are active and working. These steps install QMF control and sample tables and reload packages at the remote server.

1. Run job DSQ3EDBU. Install QMF control tables and sample tables. This job must be edited before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of ..* to * with the following command:

      ```
      ch /..*/*/ *
      ```
   c. Delete the two last lines of the file, leaving the end-of-job statement.

d. Carefully read the detailed comments in the file and change all appropriate values
e. File and run DSQ3EDBU.

2. Run job DSQ3BPKG- RELOAD QMF packages at remote server. This job must be edited before it can be run. Follow the same tasks as above in step 1, except for e, where DSQ3EBPKG instead of DSQ3EDBU will be filed and run.
3. Install the QMF LNLF at the remote DB2 UDB server. If support for a national language is being added, run DSQ3nDBU. This job must be edited before it can be run. Follow the same tasks as above in step 1, except for e, where DSQ3nDBU instead of DSQ3EDBU will be filed and run.

Installing QMF Version 7.2 for an iSeries server

The steps listed here describe a QMF server installation to a remote DB2 UDB for iSeries Version 4.4 or higher database server. The QMF server installation uses DB2 for VSE batch requester services and assumes that all connections are active and working. These steps install QMF control and sample tables and reload packages at the remote server.

Follow the steps as described in “Installation steps” on page 219 and substitute DSQ3EDBA for DSQ3EDBU in step 1. In step 3 substitute remote iSeries server for remote DB2 UDB server.

Chapter 19. Run the Installation Verification Procedure

This chapter describes the final testing of QMF, the installation verification procedure (IVP).

Before starting QMF

1. Complete all the installation and customization steps outlined in the book.
2. Start the database connection by issuing the CIRB command.
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 "Step 22—Update CICS control tables (CICS version 3 or later)" on page 72 for details.

Start and test QMF

This procedure starts the QMF for VSE 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.

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, 2000
All Rights Reserved.
IBM is a registered trademark of International Business Machines
_____________________________________________________________________________

  QMF HOME PANEL                        Query       Management    Facility
  Version 7 Release 2
                                      ******     **    **      ********* ____
  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 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, 1998
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 7.2                                      |
|     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, yout 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 is empty.

Run an IVP for NLF

Rerun the IVP, beginning with "Before starting QMF" on page 221 and start QMF by issuing a different transaction ID. Use QMF*n* where *n* is the NLID for the language, as given in Table 34 on page 202. For example, if you are installing German, the transaction ID is QMFD.

If a language that uses the double-byte character set is being tested, such as Japanese orKorean, your terminal must be double-byte enabled.

What if it did not work?

If QMF is unable to start, an error message is generated. Descriptions of some of the more common errors from running the IVP follow. If you do not find the message on this list, consult the appropriate *Messages and Codes* manual.

- **AEY9 ABEND** The database connection between the CICS partition and DB2 is not active. Issue a CIRB command.
- **G050 ABEND** The release level of GDDM being used in CICS partition does not match the version with which QMF was link-edited.
- **Gxxx ABEND** Issued by GDDM. See the *GDDM Diagnosis* manual and the *GDDM Diagnosis and Problem Determination* guide.
- **DFH1599** Region/partition size is insufficient to initialize CICS. Increase the partition size.
- **DSQ40083** GDDM ERROR ADM0962 E MAPGROUP 'DXYKIMD5' not found. SEVERITY 8 FUNCTION MSQGRP.

  The double-byte character set language feature requires a terminal that is also double-byte enabled. Before restarting QMF, ensure your terminal can display double-byte characters. If your terminal is double-byte enabled and you still have the error, check the ICCS Terminal Control Table (TCT) for proper entries.
- **DSQ51304** File DSQPNLn not found in CICS. ***CMD=HELP

  The VSAM file that contains QMF screen images is not available. Check the results of DSQ3EINS or DSQ3nINS for NLF. Also, verify teat the panel files were defined in the CICS startup and in the CICS FCT.
- **DSQI004I** Unable to load module(s) nnnnnnn

If nnnnnnn = ADMASP, verify that the GDDM product library was available when running the QMF job DSQ3ELNK. If nnnnnnn = ARIPRDI, verify that the DB2 product library was available when running QMF job DSQ3ELNK. Other modules should be available from the QMF product library.

Warning messages: QMF generates warning messages for conditions it detects while starting QMF. Your QMF trace data contains helpful information for analyzing warning messages. For example, the messages might concern the initialization of the QMF governor DSQUEGV3, or the availability of the edit exit phase DSQUECIC.

CEBR DSQD can be used to browse the temporary storage queue for warning messages.

Chapter 20. How to Maintain QMF

QMF maintenance includes adding new components and replacing existing ones. It is assumed here that QMF and all of the prerequisite products have been installed.

Adding new components

New components include new products, new versions or release of products, or additional databases.

Adding GDDM-PGF

GDDM-PGF is an optional product that can be installed after QMF. Because all GDD objects (such as charts and forms) are loaded into the ADMF file during QMF installation, no further action is required on the QMF side.

Adding QMF to another DB2 database

Repeat the procedures in "Install QMF base into DB2 database" on page 208. If national language support is needed, follow the procedures in "Install QMF base into DB2 database" on page 208.

Migrating to new releases of DB2, CICS, or GDDM

Because QMF is prelinked to specific release levels of DB2, CICS, and GDDM, it is necessary to relink-edit whenever you migrate to a new release of those products.

Binding QMF Version 7.2 packages at a remote server

QMF Version 7.2 packages must be present at the server in order for a requester installation to be able to communicate. If a complete QMF Version 7.2 new or migration installation was performed at the server, communications can be started and nothing further needs to be done.

For those servers containing QMF Version 3.3 or above, where migration is not a current option, users can run the job DSQ3BPKG. This job binds QMF Version 7.2 packages at any server specified.

Read, tailor and submit job DSQ3BPKG to perform the binds. Check the job output for error messages and rerun as necessary.

Scenario for use: Local DB2 for VSE subsystem, DB2VSE, is migrated from QMF Version 3.3 to QMF Version 7.2. QMF users in subsystem DB2VSE regularly communicate with a DB2 for VM server, SQLV61A, which contains QMF Version 3.3. The DB2 for VM DBA cannot perform a QMF migration to Version 7.2 at the VM server. In order for the QMF Version 7.2 installation in

DB2VSE to communicate with QMF on SQLV61A, job DSQ3BPKG must be run to bind packages at the DB2 for VM server.

Replacing existing components

This section describes the steps necessary to replace or reinstall QMF, and how to apply service updates to QMF.

Re-Installing QMF

1. Install the tape as described in the QMF Program Directory
2. Perform "Run the QMF installation job" on page 207 to re-install QMF.

   The initialization procedure, DSQ3INIT, does not have to be rerun unless product information was changed from the last installation. For example, if you are using another release of GDDM or installing to a different sublibrary, then the Catalog the Initialization Procedure would be followed beginning at step 4 on page 206.

   Begin the installation procedure at step 2 on page 207. When this job is run, the panel file is deleted, redefined, and reloaded. QMF charts and maps are also reloaded into the GDDM object file, ADMF.
3. Perform "Install QMF base into DB2 database" on page 208 to re-install QMF packages.

   Set the first three job steps as follows:

   ```
   // SETPARM STEP1=NO     -- CREATE QMF CONTROL TABLES IN SQL DB
   // SETPARM STEP2=YES    -- LOAD QMF PACKAGES INTO SQL DB
   // SETPARM STEP3=NO     -- LOAD QMF SAMPLES INTO SQL DB
   ```

   Continue with the remaining procedure.
4. Determine whether you need to relink-edit your products as described in "Link-edit jobs for QMF" on page 211.
5. Skip the remaining steps, because it is not necessary to tailor CICS again.

Re-installing an NLF

Follow the procedure in Install NLF starting with step 1d on page 210. DSQ3*n*DBL does not need to be run.

Determine whether you need to relink-edit your products, as described in "Link-edit jobs for QMF" on page 211. CICS does not need to be tailored again.

Applying service updates

Maintenance or service updates might need to be applied to QMF periodically. These updates are in the form of a Program Temporary Fix (PTF) tape from IBM. All QMF tapes are shipped in MSHP format for easy installation and tracking. For more information on how to apply PTFs using MSHP, see the *VSE/ESA Installation and Service* manual.

Detailed instructions for installing that specific fix accompany the PTF tape from IBM. Like most IBM products, QMF consists of phases. But, unlike most IBM products, it also consists of objects such as panels, GDDM maps, and SQL packages.

Replacing text decks or phases

This is the most common type of replacement. Apply the PTF that contains the new text deck (object) or phase. The PTF specifies to MSHP which link book to use, if necessary.

There are however, QMF objects that require your handling, as they cannot be handled automatically. The MSHP process does keep track of these changes and restores the objects to the QMF sublibrary. The PTF documentation provides details if one of the following steps is to be performed after the PTF installation.

Updating the QMF panel file

If changes are necessary to the QMF Panel file (DSQPNL*n*), the entire file does not have to be replaced. Instead, single panels are shipped using the following naming convention:

DXY*n*name.N where *n* is the NLID

DXY*n*name is the complete name of the panel.

1. Install the PTF; the panel (or panels) are restored in the QMF sublibrary.
2. Close the existing panel file using the CICS transaction, CEMT:

   ```
   CEMT SET DA(DSQPNLn) CLOSE
   ```

3. Load the panels to the VSAM panel file DSQPNL*n*. Use the following sample job to load the panel DSY*n*name to file DSQPNL*n*.

   ```
   * $$ JOB JNM=REPPANEL,DISP=D,CLASS=0
   // JOB REPPANEL  Replace panel in the QMF panel file
   // DLBL DSQPNLn,'QMF720.DSQPNLn',,VSAM.CAT=VSESPUC
   // LIBDEF *,SEARCH=(qmflib.sublib)
   // EXEC DSQCVS80,SIZE=AUTO
   * $$ SLI MEM=DXYnname.n s=qmflib.sublib
            ............
   /*
   /&
   * $$ EOJ
   ```

   Check and modify, if necessary, the following values:

   - **n** NLID. The single character that represents the language of the panel.
   - **VSESPUC** VSAM catalog name where the panel file was originally defined during QMF installation.
   - **qmflib.sublib** Library and sublibrary for QMF.

- **DXYnname** Name of the panel to be replaced. This information is provided with the PTF.
- **.....**Repeat, if necessary, the last statement for every panel being replaced by the PTF.

4. Reopen the panel file with:

```
CEMT SET DA(DSQPNLn) OPEN
```

Updating QMF GDDM maps

QMF GDDM maps can also be affected by a PTF. As with the panels, those objects are restored to the QMF sublibrary when the PTF is applied.

1. Install the PTF.
2. Modify the SETPARM statements in the QMF installation job, DSQ3EINS.

```
// SETPARM STEP1=NO     *-- DEFINE CLUSTER DSQPNLE
// SETPARM STEP2=NO     *-- LOAD DSQPNLE
// SETPARM STEP3=NO     *-- LOAD QMF CHARTS
// SETPARM STEP4=YES    *-- LOAD QMF MAPS
```

3. File and run the job. The first three job steps are skipped, and execution begins with loading the QMF maps.

NLF maps: Because GDDM maps are language dependent, your PTF might require you to change those objects as well.

1. Rerun the job DSQ3*n*INS, with the following SETPARM settings:

```
// SETPARM STEP1=NO     *-- DEFINE CLUSTER DSQPNLP
// SETPARM STEP2=NO     *-- LOAD DSQPNLP
// SETPARM STEP3=YES    *-- LOAD QMF MAP GROUPS TO ADMF
```

2. File and run the job.

Updating QMF SQL packages

If QMF SQL packages are changed with a PTF, then the packages must be loaded into each database where QMF is installed. Use the original installation job, DSQ3EDBI, to update the packages.

1. Modify the SETPARMS in the QMF installation job DSQ3EDBI, as follows:

```
// SETPARM STEP1=NO     *-- CREATE QMF CONTROL TABLES INSQLDB
// SETPARM STEP2=YES    *-- LOAD QMF PACKAGES INTO SQL DB
// SETPARM STEP3=NO     *-- LOAD QMF SAMPLES INTO SQL D B
```

2. Locate the // SETPARM DBNAME=SQLDS parameter and verify or change SQLDS to the name of the database that you are using.
3. File and run the job.
4. Repeat the procedure to load the packages into every database.

Part 4. Managing QMF

Chapter 21. Starting QMF

This chapter describes the different ways you can start QMF.

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

Any references to OS/390 mean OS/390 and z/OS.

Setting up and starting QMF on OS/390

On OS/390, QMF can be set up to run under TSO, ISPF, as a batch job, or under CICS.

Choosing an authorization ID on OS/390

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 are names that contain no more than eight characters. 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 OS390 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 OS/390 as a batch job

Allow your users to start QMF in native OS/390 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 39, 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=QMF720,S=DSN'
//*********************************************************
//*     Program load libraries
//*********************************************************
//STEPLIB  DD DSN=QMF720.SDSQLOAD,DISP=SHR
//         DD DSN=DSN720.SDSNEXIT,DISP=SHR
//         DD DSN=DSN720.SDSNLOAD,DISP=SHR
//         DD DSN=GDDM.GDDMLOAD,DISP=SHR
//*********************************************************
//*     QMF/GDDM maps
//*********************************************************
//ADMGGMAP DD DSN=QMF720.DSQMAPE,DISP=SHR
//*********************************************************
//*     Datasets 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 39. JCL to run a QMF procedure in native OS/390 batch

If you are running QMF in native OS/390, data set names that are used in QMF procedures must be fully qualified. The TSO prefix and suffix are not available in native OS/390.

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 40 lists the datasets used by TSO. If you do not know your installation's search order for REXX execs, allocate SDSQEXCE to both SYSEXEC and SYSPROC.

```
//*************************************************************
//*       DATASETS USED BY TSO                                *
//SYSPROC DD DSN=SYS2.CLIST,DISP=SHR            * CLIST Library
//        DD DSN=QMF720.SDSQCLTE,DISP=SHR
//SYSEXEC DD DSN=SYS2.EXEC,DISP=SHR
//        DD DSN=QMF720.SDSQEXCE,DISP=SHR
//SYSHELP DD DSN=SYS1.HELP,DISP=SHR
//EDT     DD DSN=&EDIT,UNIT=SYSDA,SPACE=(1668,(40,12))
```

Figure 40. 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 'QMF720.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 OS/390:

```
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 (OS/390).

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 41 on page 242 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 41. 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 OS/390

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('QMF720.SDSQEXIT,'QMF720.SDSQLOAD',
'DSN710.SDSNEXIT','DSN710.SDSNLOAD') SHR REUSE
LIBDEF ISPLLIB LIBRARY ID(DSQLLIB)
```

To allocate program libraries using the ISPF LIBDEF service, write a CLIST similar to Figure 42 on page 244. 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('QMF720.SDSQEXIT',           +
               'QMF720.SDSLOAD',             +
                 'DSN;720.SDSNEXIT',           +
                 'DSN;720.SDSNLOAD')
/*****************************************************/
/*  Allocate  QMF libraries used for GDDM            */
/*****************************************************/
ALLOC FI(ADMGGMAP) DA('QMF720.DSQMAPE')  SHR REUSE
ALLOC FI(ADMCFORM) DA('QMF720.DSQCFORM') SHR REUSE
ALLOC FI(DSQUCFRM) DA('QMF720.DSQUCFRM') SHR REUSE
ALLOC FI(ADMGDF)   DA('QMF72.CHARTLIB')  SHR REUSE
/*****************************************************/
/*  Allocate QMF product datasets                    */
/*****************************************************/
ALLOC FI(DSQPRINT) SYSOUT(Z) LRECL(133) RECFM(F B A ) BLKSIZE(1330)
ALLOC FI(DSQPNLE)  DA('QMF720.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('QMF720.SDSQPLBE')
ISPFEXE LIBDEF ISPSLIB DATASET ID('QMF720.SDSQSLBE')
ISPFEXE LIBDEF ISPMLIB DATASET ID('QMF720.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 datasets                       */
/*****************************************************/
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 OS/390. 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, as shown in Figure 47 on page 253, 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 43. 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. On OS/390 when ISPF stops, 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 (OS/390) 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 OS/390:

```
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.

Using the CICS/DB2 attachment facility

When you start QMF under CICS, you perform DB2 sign-on processing through the DB2 exit routine, DSN3@SGN. (IBM also supplies a default exit

routine.) The routine returns a list of the assigned authorization IDs if the AUTH entry provides it with the ID for the transaction in the CICS resource control table (RCT).

The plan ID and authorization IDs for a transaction ID are specified in the RCT. Use statements similar to these:

```
DSNCRCT TYPE=ENTRY,TXID=QMFE,PLAN=QMF720,AUTH=DEPT1
   DSNCRCT TYPE=ENTRY,TXID=QMFQ,PLAN=QMF720,AUTH=Q
```

Users invoking the QMFE transaction operate under the primary authorization ID DEPT1. Similarly, the QMF administrator can use the QMFQ transaction and operate with the primary authorization ID Q. If RACF is installed on your system, the authorization ID must be a valid RACF ID. The transaction IDs must also be defined in the partition control table (PCT).

The QMF programs are link-edited and bound during installation. No additional steps are necessary for CICS.

The CICS Attachment Facility uses the Resource Manager interface to access DB2 data. There is a task switch for each database fetch. To maintain an acceptable response for all users, you might want to limit the number of rows that a query can fetch.

If you choose to use the default exit routine DSN3@SGN without changes, the primary and SQL authorization IDs are the same as the ID that was obtained by the AUTH entry for the transaction in the CICS RCT.

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 OS/390

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.

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 OS/390

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 44 lists the load libraries.

```
//*************************************************************
//*     PROGRAM LOAD LIBRARIES                                *
//*************************************************************
//STEPLIB DD DSN=QMF720.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=DSN720.SDSNEXIT,DISP=SHR     * DB2 MODULES  *
//        DD DSN=DSN720.SDSNLOAD,DISP=SHR     * DB2 MODULES  *
//        DD DSN=GDDM230.SADMMOD,DISP=SHR     * GDDM MODULES *
```

Figure 44. Program load libraries for ISPF, ISPF/PDF, QMF, DB2, and GDDM

Verify GDDM data sets on OS/390: 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=QMF720.DSQMAPE,DISP=SHR   * GDDM Map Group
//ADMCFORM DD DSN=QMF720.DSQCHART,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 45. QMF/GDDM data sets

Setting up QMF to run on VM

In CMS, a user can start QMF in a batch CMS environment, or by using the DSQQMFE command either with the NUCXLOAD command or in an exec.

Note that if you use the CONCAT option on the ISPLLIB FILEDEF statement, you must also issue a GLOBAL LOADLIB DSQLDLIB.

Connecting to DB2

The DB2 for VM database program usually operates in its own virtual machine that is associated with a VM logon ID. The directories of each virtual machine can contain IUCV (inter-User Communications Vehicle) entries that allow the machines to communicate with DB2 for VM. You need to ensure the compatibility between the entries for QMF users and DB2 for VM users.

Any combination of the following cases can exist:

Case 1: The DB2 for VM directory contains IUCV ALLOW. Any other virtual machine can communicate with DB2 for VM.

Case 2: The QMF user's entry contains IUCV ANY. The QMF user can communicate with any other virtual machine, including the DB2 for VM computer.

Case 3: The QMF user's entry contains IUCV *sqlsid*, where *sqlsid* is the user ID of the DB2 for VM. The QMF user can contain this directory entry in any case, and must have it if neither case 1 nor case 2 entry applies.

Case 4: The DB2 for VM directory contains an IUCV *IDENT control statement to identify which resources it manages, and whether the resources

are local or global. A local resource can be accessed only by QMF users on the same processor. A global resource can be accessed by QMF users on local or remote processors.

If your installation requires the use of QMF in different databases, you must install QMF into each unique DB2 database. Each database contains the following items:

- QMF control tables
- QMF DB2 for VM packages
- QMF sample tables and queries
- QMF views (stem tables)

Initializing the QMF session: The DB2 for VM user ID must be the same as the VM logon ID during the QMF session initialization. QMF connects implicitly during the session.

DB2 for VM considerations

QMF supports DATE, TIME, and TIMESTAMP data types so that users can use local date/time exit routines. For QMF to use a local date/time exit, the text files that contain the date/time exits, RIUXDT and ARIUXTM, must be placed on a disk that is accessible to QMF when QMF is started.

The QMF DCSS includes the ARIRVSTC text file. The QMF DCSS must be rebuilt using the DSQ2ESEG EXEC if this file is changed by PTFs applied to DB2 for VM or a new level of DB2 for VM.

GDDM considerations on CMS

When the QMF DCSS is built, it includes the GDDM interface code. If you run GDDM from a DCSS, you do not need to access a GDDM disk or GDDM TXTLIBs, and you can remove the lines in the invocation exec that refer to GDDM.

If you do not have a GDDM in a DCSS, you must access the GDDM TXTLIBs and perform the necessary FILEDEFs. If you want to change the release of GDDM that QMF uses, you must rebuild the QMF share segment using the DSQ2ESEG EXEC.

Setting up and starting QMF on CMS

The following examples show how to start and pass parameters to QMF operating independently of ISPF. The first two statements for CMS 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 CMS:

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

- Starting from an 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

  To run QMF independently of ISPF, use either of the following commands:

```
DSQQMFE dcssname(DSQSBSTG=n1,...)
DSQQMFE DSQSBSTG=n1,...
```

  The DSQSRUN parameter as specified in this example results in the following QMF command:

```
RUN Q.IPROC(&TABLE=Q.STAFF
```

Setting up QMF to run under ISPF

You can let users start QMF using ISPF services.

- Add QMF to an initial dialog of ISPF.
- Replace the initial dialog with one that starts QMF directly.
- Create an exec to start QMF as a program dialog.

You can use any of the previous methods to start the other methods. For example, you can run an initial dialog from an exec.

To run QMF under ISPF, you must start the QMF program dialog using the ISPF SELECT service. When a CMS command is used, the results can be unpredictable.

Restrictions:

1. You cannot run QMF as a command dialog.
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 on CMS

In the definition shown below, Option 2 was added for reaching QMF through an exec.

```
)BODY
%---------------------- MASTER APPLICATION MENU -----------------
%SELECT APPLICATION ===>_OPT   +
%                                              +USERID    -
%                                              +TIME      -
% 1 +SPF   -SPF PROGRAM DEVELOPMENT FACILITY    +TERMINAL  -
% 2 +QMF   -QMF --English                       +FUNCTION KEY -
% 3 +QMFU  -QMF --Uppercase
% 4 +QMFK  -QMF --Japanese
% 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(ISP@PRIM)NEWAPPL'
             2,'PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQIROW=150)'
             3,'PGM(DSQQMFU) NEWAPPL(DSQU) PARM(DSQIROW=150)'
             4,'PGM(DSQQMFK) NEWAPPL(DSQK) PARM(DSQIROW=150)'
           /*                              */
           /* ADD OTHER APPLICATIONS HERE */
           /*                              */
              P,'PANEL(ISPOPT)'
              X,'EXIT'
            ' ',' '
              *,'?' )
)END
```

Figure 46. ISPF on CMS sample Master Application menu

If you are using an NLF: You can change the definition of the Master Application menu to allow users to pick the language environment for their QMF sessions. Figure 46 is an example in which users have a choice of beginning a QMF session in English (option 2), Uppercase (option 3), or Japanese (option 4). The TRANS function in the)PROC section of the panel definition transforms options 2, 3, and 4 into the operand portions of an ISPF command that is executed like ISPSTART. The command invokes the appropriate QMF module (DSQQMFE, DSQQMFU, or DSQQMFK), and passes it the value 160 for the DSQSIROW parameter.

Tip: The direct menu approach can start QMF as much as four times faster than the exec approach. If you allocate all user resources through CMS logon procedures, then the exec that you create for the menu option has no resources to allocate. The single function, starting QMF, can be run without an exec.

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 an exec.

- Without an exec:

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

- With an exec:

  ```
  ISPSTART CMD(exec_name) NEWAPPL
  ```

  where *exec_name* is the name of the exec 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, as shown in Figure 47, 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 47. 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. On CMS, when ISPF stops, control is returned to the next line in the exec.

Examples of starting QMF under ISPF

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

- Starting ISPF from an exec 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 an exec operating within ISPF on CMS:

  ```
  ISPEXEC SELECT PGM(DSQQMFE) NEWAPPL(DSQE) PARM(DSQSDCSS=QMF)
  ```

  This statement passes the name QMF for the QMF program segment.

- 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 exec 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
```

Verify QMF data files on VM

The following list of data files is used by QMF in CMS. These files are allocated according to the recommended sizes in the DSQ2EINV exec. If you want to allocate them differently, you must change the invocation exec.

DSQDEBUG
QMF trace dump output

DSQDEBUG cannot be allocated to a disk by using the shared file system (SFS)

DSQPRINT
Print data output

DSQSPILL
Spill data file

DSQPNLE
QMF panel file

ISPLLIB
FIledef for QMF library; contains the QMF programs

Verify program modules on CMS

The DB2 for VM database, QMF's program segments, ISPF's shared segments (if used), and GDDM's shared segments or product text libraries must be available before starting QMF.

Setting up and starting QMF on VSE

QMF is provided as a standard CICS transaction.

Starting QMF from the VSE/ESA function selection menu

You can add the invocation of QMF to the VSE/ESA Function Selection menu. Use these procedures to add QMF to your VSE system:

1. Create a new application profile for QMF.
2. Add or change the QMF application profile.
   a. Ensure that the CODE field specifies QMF as a NON-CONVERSATIONAL transaction with data.
   b. Specify the name of the QMF transaction ID QMF*n* in the ACTIVATE field where *n* is the NLF ID. The QMF transaction ID for English is QMFE.
   c. Specify any QMF program parameters that you want to use in the DATA field.
3. Add the new QMF application profile to a selection panel.

   After you add QMF to the VSE/ESA function selection menu, the menu might look like this:

```
Enter the number of your selection and press the ENTER key:
1. Operations
2. Problem Handling
3. Program Development
4. Command Mode
5. CICS -supplied transactions
6. CIRB -start SQL connection
7. ISQL- Interactive SQL Facility
8. QMF- Query Management Facility
```

For more information on the function selection menu, see the

Connecting CICS and DB2 for VSE

This section is common to all methods of starting QMF on VSE.

Establishing the connection between CICS and DB2 VSE sets up DB2 online support and allows users at CICS terminals to communicate with the DB2 VSE application server through CICS.

CIRB is usually run as part of the job that starts the CICS partition. To run CIRB manually:

1. Start the DB2 VSE application server in multiple user mode, according to instructions in *DB2 Server for VSE System Administration.*
2. Run the CICS CIRB transaction once for each CICS partition where QMF is installed. You can run the transaction in one of three ways:
   - By starting it from any CICS terminal

- By starting it from the VSE console
- Automatically when CICS comes up, if you make the appropriate changes in the CICS startup facilities

Starting QMF from a CICS application

QMF can interact with existing QMF applications that you have at your site. You can use the EXEC CICS START command with the transaction ID QMFn to start a QMF session from within a CICS application. An example of the command is shown in the following example. Replace the n symbol with the NLID from the Table 1 on page xiv.

```
EXEC CICS START
TRANSID('QMFn') FROM('M=B,I=START_PROC,UID=Q/QMF') TERMID('MYT5')
```

The command starts a noninteractive QMF session, connects QMF to DB2 using a user ID of Q and a password of QMF, then runs a QMF procedure named START_PROC.

Use the same rules for passing QMF program parameters that you use to start QMF from a cleared CICS screen, as shown in "Starting QMF from a cleared CICS screen" on page 257. You can use any QMF program parameter in a CICS application.

A terminal ID (TERMID) is required for an interactive session (when DSQSMODE = I), and optional for a noninteractive session (when DSQSMODE = B). If the terminal ID specifies the terminal where the calling CICS application is running, the QMF session starts when the CICS application finishes. If you do specify a terminal ID, the terminal must be available. Also ensure that the ID is defined as either a local terminal or a remote terminal on the system from which the START command is issued.

If you do not know the TERMID, issue the EXEC CICS ADDRESS EIB(xxx) parameter to retrieve it.

Starting a noninteractive session

You might choose to run a nonineractive QMF session to conserve resources. Use a value of B for the DSQSMODE parameter and make sure that you use the DSQSRUN parameter to pass the name of an initial procedure to perform the necessary QMF tasks. Use the DSQSUSER parameter to ensure that you connect to the database using the appropriate SQL authorization ID and password.

If you do not specify a terminal ID, the QMF session runs without a terminal.

Starting an interactive session

You can also start an interactive QMF session from within a CICS application. For example, the CICS application might be a menu application that allows

users to start QMF from a menu of other products. A terminal ID is required to start an interactive session. Because the session runs interactively, you do not need to supply an initial procedure that runs when QMF starts, nor do you need to supply a value for the DSQSMODE parameter. If you want to connect to DB2 explicitly, supply values for the DSQSUSER parameter; otherwise, QMF connects to DB2 using the default VSE operator ID and password that are defined in the system catalog.

Starting QMF from a cleared CICS screen

QMF runs as a conversational transaction in CICS, and is defined in CICS resource tables during QMF installation. You can start QMF by issuing the QMF*n* transaction from a cleared CICS screen, as shown here:

```
QMFE B=600000,F=200,L=YES
```

The letters following QMFE represent abbreviated forms of some of the QMF program parameters that you can use to customize the behavior of a QMF session. For example, the values shown here start an interactive English QMF session, retrieve 200 rows of data before displaying the first screen of the report, and create active extra storage for report data when the amount of data retrieved into GETVIS storage reaches 600,000 bytes.

You can specify the parameters in any order on the QMF*n* transaction. Ensure that you meet the following requirements:

- Specify each value in a *parameter_name=value* format. You can use the short form if the parameter has one.
- Specify only one value for each parameter.
- Enter a blank, a comma, or both after each value.
- Capitalize all letters in the parameter string.

QMF uses default values in Chapter 22, “Customizing Your Start Procedure”, on page 259 for parameters that are not entered following the QMF*n* transaction. The values that you supply remain effective throughout the QMF session, except for the parameter that specifies the level of detail in the trace data. Users can change this trace parameter directly from their profiles using the SET PROFILE command.

Chapter 22. 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 the *Developing QMF Applications* manual.

Choosing the right amount of virtual storage 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 OS/390 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 OS/390 and 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 native OS/390, TSO, or ISPF 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 30, “Controlling QMF Resources using a Governor Exit Routine”, on page 563.

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 OS/390 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 OS/390, TSO commands, REXX, ISPF or other non-QMF services during QMF sessions. Those users who do use an OS/390 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 an OS/390 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 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 35 on page 264).
   - 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 35. 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 36 shows a sample calculation for a spill file.

Table 36. 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.

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 263 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 the *QMF Reference* manual 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 692 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 692.

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. See

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 692 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 692.

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. See

Customizing your start procedure on VM

Follow these instructions for customizing your start procedure on VM.

Naming the program segment

Use *dsqsdcss* or *dcssname* to name the program segment. The suggested program segment name and default value is:

```
QMF720E
```

dcssname

The syntax of *dcssname* is still supported in QMF:

1. This parameter is optional in the PGM form of the ISPSTART command ISPSTART PGM(DSQQMFE) provided that the default DCSS name is used.

2. In the DCSS form of the command ISPSTART DCSS(*dcssname*), a DCSS name must be specified.
3. If QMF is not running as an ISPF dialog, and DSQQMFE *dcssname*(B=*n1*,...) is used to start QMF, the parameter is optional.

DSQSDCSS

You can add DSQSDCSS to the list of parameters to be passed when starting QMF. For example:

```
DSQSDCSS=QMFNEW
```

DSQSDCSS supports the callable interface for QMF.

Setting default start values using the REXX program DSQSCMDn

Specify default values for the program parameters with an initialization program. IBM supplies the REXX program DSQSCMD*n* for this purpose. DSQSCMD*n* can change the default program parameter values and can execute across environments.

The parameter values you specify when you start QMF override the values set in the REXX program DSQSCMD*n*. The parameter values you specify when a workstation session is started override the values set in DSQSCMD*n*.

DSQSCMD*n* is valid as a start function keyword on the START command when QMF is started from an application program using the callable interface.

You must use the REXX program method if you want to run programs in SAA environments that use the callable interface without changing the programs.

For more information on the START command and the SAA callable interface, see the *Using QMF* and *Developing QMF Applications* manuals.

For CMS, QMF calls the REXX program DSQSCMDE to provide values for the program parameters. This IBM-supplied program supplies default values; by adjusting these values, you can tailor the QMF environment for your installation.

Use NULL if you do not want to provide a parameter value in DSQSCMDE.

Program parameters for VM

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.

Choosing the right amount of virtual storage for each user: Each QMF CMS region requires at least 4.5 MB of virtual storage. Additional storage generally provides improved performance since QMF is able to keep more data records in virtual storage.

Performance tradeoffs: Use the DSQSPILL parameter to provide users with a spill file, which is disk 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 still complete report processing. 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 CMS 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 VM system services, CMS commands, REXX, ISPF or other non-QMF services during QMF sessions. Those users who do use a VM system service or a CMS 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 CMS transactions, QMF lets you allocate a spill file, which is extra storage used when a user's storage is full.

You can reset the DSQSPILL parameter to NO to deactivate the spill file:

```
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 and the data object is complete
- Storage you defined for the spill file is full.

Allocating a spill file for CMS users: You can allocate a spill file through a FILEDEF statement:

```
FILEDEF DSQSPILL DISK DSQSPILL DATA T (LRECL 4096 RECFM F PERM'
```

The statement:

- Allocates the spill file to the T disk. The T disk can be a temporary disk. The spill file cannot be allocated to a disk that is used in the CMS shared file system (SFS).
- Specifies the DSQSPILL file with fixed-length records, one record for each block. Records must always be unblocked (a block is the size of a VM page: 4,096 bytes).

Estimating the space required for a spill file: To accommodate QMF's storage requirements, ensure the CMS 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 CMS virtual 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.
   - 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.

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.

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: 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 277 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 the *QMF Reference* manual 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. DB2 maintains an EDM pool to service its requesters. While a data object is incomplete, the requester contends with all other requesters for EDM resources.

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 stops retrieving rows or terminates when storage is full.

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
```

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 692 for more information.

Controlling initial activities during a session

This section explains program parameters that help you control initial QMF activities, such as:

- Specifying a location for the connection to the database
- Starting a noninteractive session
- Running an initial procedure that does the predetermined amount of work defined in the procedure and then exits QMF

DSQSDBNM (specifying the location to connect to when starting QMF)

Parameter name

DSQSDBNM

Short form

D

Valid values

Any valid database name

Default

The default database in use by the subsystem

You can use DSQSDBNM to specify the location to which you are initially connected for a QMF session. This location can be a remote database. You can specify DSQSDBNM in all operating environments.

If you are setting up for a remote unit of work: The maximum length in characters of the DSQSDBNM value depends on the type of the application requester that initiates the remote unit of work connections. The lengths for each requester type are shown in Table 37.

Table 37. Maximum Length of DSQSDBNM Value Based on Requester Type on VM

Requester Type	Maximum Length
DB2 for VM	18

DSQSMODE (specifying an interactive or noninteractive QMF session)

Parameter name

DSQSMODE

Short form

M

Valid values

B (noninteractive) or I (interactive)

Default

I (B if started through the callable interface)

Some query and report-writing tasks users need to perform might not require interaction with QMF. For example, a salesperson might use the same QMF

procedure every few days to query a set of tables for account status. Although the data changes, the procedure and tasks required to access the data remain the same.

Using the QMF program parameter DSQSMODE, you can save resources and time by starting a noninteractive session to perform your QMF work. Your terminal is then free for you to do other work while the transaction is running.

Use a value of B to start a noninteractive session:

```
DSQQMFn M=B,I=STARTPROC
```

Because a noninteractive session displays no QMF panels, use the DSQSRUN (I) parameter to run an initial procedure that does the required QMF work and exits the program.

Use the DSQSDBNM parameter to specify an ID and password for the database connection if you do not want to use the default database location.

DSQSRUN (naming a procedure to run when QMF starts)

Parameter name
DSQSRUN

Short form
I

Valid values
Any valid procedure name (see the *QMF Reference*) manual.

Default
No initial procedure is run

Use the DSQSRUN parameter to pass the name of a QMF procedure that runs as soon as QMF starts. In a noninteractive session, use this procedure to perform the QMF work you need to do, then exit the program.

For example, to run an initial procedure named STARTPROC, enter:

```
DSQQMFn I=STARTPROC
```

Qualify the procedure name with the SQL authorization ID of its owner if other users are using it to start QMF. For example, if user JONES owns the STARTPROC procedure, enter:

```
DSQQMFn I=JONES.STARTPROC
```

When you pass the name of an initial procedure, QMF issues a RUN PROC command, which runs the procedure you name.

Note: QMF does not allow blanks in the user ID and procedure syntax. For example, QMF does not recognize:

```
DSQQMFn I=JONES. STARTPROC
```

To use a procedure name with an imbedded blank, you must enclose the name in quotes:

```
DSQQMFn I=JONES.'START PROC'
```

Use DSQSRUN to help you automate noninteractive QMF work and allow users to perform interactive QMF work within the confines of a predefined procedure.

Running an initial procedure noninteractively: To conserve resources, you can run a procedure noninteractively by using a value of B for the DSQSMODE parameter and naming a procedure using the DSQSRUN parameter. For example, suppose that every Monday morning you need to produce an inventory status report. Each Sunday night you need to run a query that retrieves data from the same columns of a table called INVENTORY. Your query might look like the following sample, INVENTORY__QUERY:

```
SELECT * FROM INVENTORY
WHERE STOCK < 20
```

The procedure you use to run this query and print the status report might look like the following procedure, INVENTORY__PROC:

```
RUN QUERY INVENTORY__QUERY
PRINT REPORT
EXIT
```

The procedure includes an EXIT command because, when QMF is running noninteractively, no user is present to end the QMF session. EXIT ends the QMF session and frees the resources being held by QMF. Always use an EXIT command in an initial procedure that runs noninteractively.

Because the tasks involved in creating the report do not change (only the data changes), you can use the DSQSRUN parameter to query the INVENTORY table Sunday night and print the report so you can have it Monday morning:

```
DSQQMFn I=INVENTORY_PROC,M=B
```

Performing interactive QMF work with an initial procedure: You can use an initial procedure in an interactive QMF session to predefine data access tasks for end users, allowing them to access only the data they need. For example, suppose a QMF end user has the responsibility of producing an inventory status report every Monday morning. The user might know the value that indicates low stock, but may not know exactly how to produce the status report. You could insert a variable in the query so that the user would only need to enter the value that indicates low stock. We could call this query INVENTORY__QUERY:

```
SELECT * FROM INVENTORY
WHERE STOCK < &LOWSTOCK
```

Because the user might want to view the data before printing it, the INVENTORY__PROC procedure might not include the EXIT command:

```
RUN QUERY INVENTORY__QUERY
```

You can then use the DSQSRUN parameter without specifying the DSQSMODE parameter, so that you start an interactive session for the user:

```
QMFn I=INVENTORY__PROC
```

The INVENTORY__PROC procedure prompts the user for the `&LOWSTOCK` variable value. The *QMF Reference* manual explains variables in more detail.

As soon as the user provides the value, QMF displays the report and the user can then view the report and issue a QMF PRINT command to print it.

For interactive sessions, instruct users to enter EXIT on the command line when they are finished viewing the report. The initial procedure runs repeatedly until an EXIT command is issued. Pressing the End function key from the report panel reruns the initial procedure; it does not display the QMF Home panel.

Additionally, when you use the DSQSRUN parameter, make sure that the DSQEC__RERUN__IPROC global variable is set to 0 and that the current object is not the QMF Home panel. The *Developing QMF Applications* manual provides more information on this global variable, as well as information about how to write procedures that help users perform QMF activities specified in predefined procedures and applications.

Passing variable values to an initial procedure: When you supply the name of an initial procedure on the DSQSRUN parameter, you can also supply values for variables contained in the procedure. You can specify one or more variables and their values following the procedure name on the DSQSRUN parameter.

Follow these rules when you specify variables for DSQSRUN:

- Put parentheses around the variable parameter list, as shown in the examples in this section.
- Precede the variable name with an ampersand, and make sure the string is in a *variable__name=value* format.
- Ensure the combined total of characters for the procedure name and the variable parameter list is 98 characters or less.
- Separate the variable parameter specifications using a single comma, one or more blanks, or a combination of a comma and blanks.

Table 38 lists environments and the number of ampersands required to use a variable in each environment.

Table 38. Required number of ampersands preceding program variables

Environment	Number of additional ampersands	Example
CMS with ISPF	1	&&variable=value
CMS without ISPF using CLIST	2	&&&variable=value
CMS with ISPF using CLIST	3	&&&&variable=value

When you specify the name of an initial procedure, QMF issues a RUN PROC command that runs the procedure. When you use variables in your procedure, the values you supply for these variables must conform to the syntax used for passing variables on a RUN command. For information about this syntax, see the *QMF Reference* manual.

For example, suppose you frequently need two pieces of information about employees in your organization. One piece of information is the name of the employee, and the other varies. You could define a query that includes NAME and uses a variable for the other column. Figure 48 shows an example query and procedure. The figure also shows how to pass a value for the variable when you enter the DSQSRUN parameter, and shows the RUN PROC command that QMF issues.

```
Query (named JONES.QUERY2)
        SELECT NAME, &COL FROM Q.STAFF
Procedure (named JONES.PROC2)
        RUN QUERY JONES.QUERY2 (&&COL=&COL
DSQSRUN parameter
        QMFn I=JONES.PROC2(&COL=YEARS)
Resulting RUN command
        RUN PROC JONES.PROC2 (&COL=YEARS)
```

Figure 48. Passing a QMF column name using DSQSRUN

Figure 49 on page 287 shows a similar example, but instead of passing one column name to the procedure, it allows you to pass several, which return the employee's name, the department, and the employee's salary.

```
Query (named JONES.QUERY3)
        SELECT &COLS FROM Q.STAFF
Procedure (named JONES.PROC3)
        RUN QUERY JONES.QUERY3 (&&COLS=&COLS
DSQSRUN parameter
        QMFn I=JONES.PROC3(&COLS=((DEPT,NAME, SALARY))
Resulting RUN command
        RUN PROC JONES.PROC3(&COLS=((DEPT,NAME,SALARY)))
```

Figure 49. Passing several QMF column names using DSQSRUN

The next four examples show how to pass information you normally supply after the WHERE keyword in a query. (See the *QMF Reference* manual for more information about the WHERE keyword.)

These examples contain character strings. Special syntax is required due to the way QMF evaluates the values when it processes the RUN PROC command. Special characters (comma, blank, parentheses, quotes, apostrophe or single quote, and equal sign) can also be included in the string as shown.

For example, if you need to know the names and employee numbers of all the managers in your organization, you could run a query like the one in Figure 50. When you pass the character string `MGR` on the DSQSRUN parameter, be sure to enclose the value in single quotes.

```
Query (named JONES.QUERY4)
        SELECT JOB, NAME, ID FROM Q.STAFF WHERE JOB=&JOB
Procedure (named JONES.PROC4)
        RUN QUERY JONES.QUERY4 (&&JOB=&JOB
DSQSRUN parameter
        QMFn I=JONES.PROC4(&JOB='MGR')
Resulting RUN command
        RUN PROC JONES.PROC4 (&JOB='MGR')
```

Figure 50. Passing a string within single quotes using DSQSRUN

Figure 51 on page 288 shows how to pass variable values that contain commas. Enclose the value `SAN JOSE, CA` in single quotes because it contains a comma.

```
Query (named JONES.QUERY5)
        SELECT * FROM Q.APPLICANT WHERE ADDRESS+&CITY
Procedure (named JONES.PROC5)
        RUN QUERY JONES.QUERY5 (&&CITY=&CITY
DSQSRUN parameter
        QMFn I=JONES.PROC5(&CITY='SAN JOSE,CA')
Resulting RUN command
        RUN PROC JONES.PROC5 (&CITY='SAN JOSE,CA')
```

Figure 51. Passing a comma within a string using DSQSRUN

Figure 52 shows how to pass variable values that contain single quotes (for example, an apostrophe in a name). When you pass the value on the DSQSRUN parameter, be sure to enclose the value in single quotes and use two single quotes for the apostrophe instead of one.

```
Query (named JONES.QUERY6)
        SELECT * FROM Q.STAFF WHERE NAME=&NAME
Procedure (named JONES.PROC6)
        RUN QUERY JONES.QUERY6 (&&NAME=&NAME
DSQSRUN parameter
        QMFn I=JONES.PROC6(&NAME='O''BRIEN')
Resulting RUN command
        RUN PROC JONES.PROC6 (&NAME='O''BRIEN')
```

Figure 52. Passing an apostrophe as part of a string using DSQSRUN

Figure 53 shows how to pass values for variables in two different parts of the query.

```
Query (JONES.QUERY7)
        SELECT * FROM Q.STAFF WHERE DEPT IN &DEPT AND JOB=&JOB
Procedure (named JONES.QUERY7)
        RUN JONES.QUERY7 (&&DEPT=&V1 &&JOB=&V2
DSQSRUN parameter
        QMFn I=JONES.PROC7(&V1=(((10,38))) &V2='MGR')
Resulting RUN command
        RUN PROC JONES.PROC7(&V1=(((10,38))) &V2='MGR')
```

Figure 53. Passing multiple variable parameters and values using DSQSRUN on VM

DSQSDBCS (Setting printing for double-byte character set data)

Parameter name

DSQSDBCS

Short form
K
Valid values
YES or NO
Default
NO

If you use the Uppercase or Japanese NLF, you might need to print double-byte character set (DBCS) data. You can set the DSQSDBCS program parameter to YES to print DBCS data from non-DBCS terminals.

For example, suppose a user with an IBM 3279 display terminal needs to print a table (DBCSTABLE) whose nonnumeric columns contain DBCS data. The following statement starts the Uppercase NLF from a cleared CMS screen and allows the user to print DBCSTABLE using a command such as `PRINT DBCSTABLE (PRINTER=DBCSPRT`.

```
QMFU K=YES
```

For more information on how to establish a GDDM nickname for the DBCSPRT printer, see Chapter 26, "Enabling Users to Print Objects", on page 423.

Customizing your start procedure on VSE

Follow these instructions to customize your start procedure on VSE.

Program parameters for VSE

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 GETVIS storage. This section explains QMF program parameters that help you customize:

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

DSQSBSTG (adjusting GETVIS storage used for report data)

Parameter name
DSQSBSTG
Short form
B
Valid values
From 0 to 99,999,999 bytes
Default
500,000 bytes

In VSE, to produce reports and temporarily store data, QMF uses GETVIS storage, which is virtual storage within the CICS partition. VSE/ESA 1.3 limits GETVIS storage according to the partition size you define for CICS. To ensure each user has enough storage for QMF queries and reports, first adjust the CICS partition size according to the number of QMF users and the size and complexity of reports they are creating.

After you size the CICS partition, use the DSQSBSTG parameter to specify the maximum amount of GETVIS storage QMF uses to run queries and produce reports. Specify the storage amount in bytes. The user can specify the GETVIS storage from a cleared CICS screen.

For example, the following command starts QMF from a cleared CICS screen and specifies that a maximum of 0.8 MB of GETVIS storage can be used to store the user's report data:

```
QMFn B=800000
```

Choosing the right amount of GETVIS storage for each user

QMF needs a minimum amount of GETVIS storage to display a report in the default format. This minimum is between 15,000 and 31,000 bytes (15 to 31 KB) depending on how the storage in your CICS partition is distributed. Set DSQSBSTG to 0 when you want QMF to use the minimum value for GETVIS storage.

The default value of 0.5 MB can accommodate most QMF transactions. However, the amount of virtual storage needed varies for individuals using a report format other than the default. Users working with very large reports may need up to 1 MB or more of virtual storage. See the *QMF Reference*for information on report formatting options.

Important: QMF requires a minimum of 15 MB GETVIS storage for up to 20 users (24 MB total virtual storage for the partition). When you increase a user's GETVIS storage using the DSQSBSTG parameter or when you add more QMF users, make sure you increase the value of the CICS ALLOC parameter so that each user has enough GETVIS storage to run queries and produce reports. A QMF transaction could time out waiting for storage to become available.

Performance tradeoffs

Use the DSQSPILL parameter to provide users with a spill file; if the spill file is full, QMF continues to retrieve data into GETVIS storage in amounts specified by the DSQSBSTG parameter. If you use too low a value for DSQSBSTG, performance will be poor even if you use a spill file, because QMF must return to the database several times to retrieve the requested data. Consider using a governor exit routine to limit rows retrieved from the database so that less GETVIS storage is used for queries and reports.

DSQSPILL (acquiring extra storage)

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

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.

Set the DSQSPILL parameter to YES to activate the spill file:

```
QMFn L=YES
```

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 data in the spill file exceeds the maximum of 32 ,767 rows (each row holds 4 KB of data)

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 CICS storage. In CICS, temporary storage for the spill file is limited to 32,767 buffers that are each 4 KB.

To accommodate QMF's storage requirements, ensure the CICS temporary storage file DFHTEMP 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.
   - 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.

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.

Solving some spill file problems

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.

DSQSSPQN (specifying the name of the CICS spill storage)

Parameter name

DSQSSPQN

Short form

(no short form)

Valid values

Any name that follows CICS naming conventions for queues

Default

DSQSnnnn (nnnn is the CICS terminal ID)

When you use a spill file, you can also specify a name for the CICS temporary storage queue to use for QMF spill data. For example, to specify the name MYDATA:

```
QMFn DSQSSPQN=MYDATA
```

If you start a noninteractive QMF session from within a CICS application and choose not to specify a CICS terminal ID, you need to code the DSQSSPQN parameter. You must explicitly specify a value for DSQSSPQN, or QMF does not start.

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 291 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 the *QMF Reference* manual 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. DB2 maintains an EDM pool to service its requesters. While a data object is incomplete, the requester contends with all other requesters for EDM resources.

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. When you plan your values for DSQSBSTG and DSQSIROW, remember that in CICS, QMF could time-out while waiting for storage to become available.

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 692 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 692.

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.

Controlling initial activities during a session

This section explains program parameters that help you control initial QMF activities, such as:

- Specifying a location for the connection to the database
- Starting a noninteractive session
- Running an initial procedure that does the predetermined amount of work defined in the procedure and then exits QMF

DSQSUSER (Connecting to the database)

Parameter name
DSQUSER

Short form
UID

Valid values
ID and password that conform to CONNECT command rules

Default
3-character VSE operator ID and password defined in the DB2 system catalog

When a user starts QMF, DB2 uses an authorization ID to determine whether the user is authorized to connect to the database. DB2 uses this same ID to determine a user's authorization to access objects and perform database activities.

You can use the DSQSUSER parameter to provide DB2 with an authorization ID and password to use for the database connection. For example, the following command connects user JONES, who has a password of MYPW:

```
QMFn UID=JONES/MYPW
```

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 VSE DB2. For more information about these rules, see the DB2 Server for VSE and VM SQL Reference.
- The SQL authorization ID and password must both be in the DB2 system table SYSTEM.SYSUSERAUTH.

If you do not supply an SQL authorization ID and password, DSQSUSER defaults to the three-character VSE operator ID and password defined in the DB2 system catalog. You can issue the following SQL statements from the SQL query panel at any time during the QMF session to determine the ID that DB2 is currently using for database authorization:

```
SELECT DISTINCT USER FROM Q.ORG
```

If you supply a user ID, but no password, QMF displays an error message. The password you supply does not have to be identical to the pasword associated with the VSE logon ID.

DSQSMODE (specifying an interactive or noninteractive QMF session)

Parameter name

DSQSMODE

Short form

M

Valid values

B (noninteractive) or I (interactive)

Default

I (B if started through the callable interface)

Some query and report-writing tasks users need to perform might not require interaction with QMF. For example, a salesperson might use the same QMF procedure every few days to query a set of tables for account status. Although the data changes, the procedure and tasks required to access the data remain the same.

Using the QMF program parameter DSQSMODE, you can save resources and time by starting a noninteractive session to perform your QMF work. Your terminal is then free for you to do other work while the transaction is running.

Use a value of B to start a noninteractive session:

```
DSQQMFn M=B,I=STARTPROC
```

Because a noninteractive session displays no QMF panels, use the DSQSRUN (I) parameter to run an initial procedure that does the required QMF work and exits the program.

Use the DSQUSER parameter to specify an ID and password for the database connection if you do not want to use the default VSE operator ID and password.

DSQSRUN (naming a procedure to run when QMF starts)

Parameter name
DSQSRUN

Short form
I

Valid values
Any valid procedure name (see the *QMF Reference*) manual.

Default
No initial procedure is run

Use the DSQSRUN parameter to pass the name of a QMF procedure that runs as soon as QMF starts. In a noninteractive session, use this procedure to perform the QMF work you need to do, then exit the program.

For example, to run an initial procedure named STARTPROC, enter:

```
DSQQMFn I=STARTPROC
```

Qualify the procedure name with the SQL authorization ID of its owner if other users are using it to start QMF. For example, if user JONES owns the STARTPROC procedure, enter:

```
DSQQMFn I=JONES.STARTPROC
```

When you pass the name of an initial procedure, QMF issues a RUN PROC command, which runs the procedure you name.

Note: QMF does not allow blanks in the user ID and procedure syntax. For example, QMF does not recognize:

```
DSQQMFn I=JONES. STARTPROC
```

To use a procedure name with an imbedded blank, you must enclose the name in quotes:

```
DSQQMFn I=JONES.'START PROC'
```

Use DSQSRUN to help you automate noninteractive QMF work and allow users to perform interactive QMF work within the confines of a predefined procedure.

Running an initial procedure noninteractively: To conserve resources, you can run a procedure noninteractively by using a value of B for the DSQSMODE parameter and naming a procedure using the DSQSRUN parameter. For example, suppose that every Monday morning you need to produce an inventory status report. Each Sunday night you need to run a query that retrieves data from the same columns of a table called INVENTORY. Your query might look like the following sample, INVENTORY__QUERY:

```
SELECT * FROM INVENTORY
WHERE STOCK < 20
```

The procedure you use to run this query and print the status report might look like the following procedure, INVENTORY__PROC:

```
RUN QUERY INVENTORY_QUERY
PRINT REPORT (QUEUENAME=Q1, QUEUETYPE=TS)
EXIT
```

The procedure includes an EXIT command because, when QMF is running noninteractively, no user is present to end the QMF session. EXIT ends the QMF session and frees the resources being held by QMF. Always use an EXIT command in an initial procedure that runs noninteractively.

Because the tasks involved in creating the report do not change (only the data changes), you can use the DSQSRUN parameter to query the INVENTORY table Sunday night and print the report to the storage queue named Q1, so you can have it Monday morning:

```
QMFn I=INVENTORY__PROC,M=B
```

Performing interactive QMF work with an initial procedure: You can use an initial procedure in an interactive QMF session to predefine data access tasks for end users, allowing them to access only the data they need. For example, suppose a QMF end user has the responsibility of producing an inventory status report every Monday morning. The user might know the value that indicates low stock, but may not know exactly how to produce the status report. You could insert a variable in the query so that the user would only need to enter the value that indicates low stock. We could call this query INVENTORY__QUERY:

```
SELECT * FROM INVENTORY
WHERE STOCK < &LOWSTOCK
```

Because the user might want to view the data before printing it, the INVENTORY__PROC procedure might not include the EXIT command:

```
RUN QUERY INVENTORY__QUERY
```

You can then use the DSQSRUN parameter without specifying the DSQSMODE parameter, so that you start an interactive session for the user:

```
QMFn I=INVENTORY__PROC
```

The INVENTORY__PROC procedure prompts the user for the &LOWSTOCK variable value. The *QMF Reference* manual explains variables in more detail.

As soon as the user provides the value, QMF displays the report and the user can then view the report and issue a QMF PRINT command to print it.

For interactive sessions, instruct users to enter EXIT on the command line when they are finished viewing the report. The initial procedure runs repeatedly until an EXIT command is issued. Pressing the End function key from the report panel reruns the initial procedure; it does not display the QMF Home panel.

Additionally, when you use the DSQSRUN parameter, make sure that the DSQEC__RERUN__IPROC global variable is set to 0 and that the current object is not the QMF Home panel. The *Developing QMF Applications* manual provides more information on this global variable, as well as information about how to write procedures that help users perform QMF activities specified in predefined procedures and applications.

Passing variable values to an initial procedure: When you supply the name of an initial procedure on the DSQSRUN parameter, you can also supply values for variables contained in the procedure. You can specify one or more variables and their values following the procedure name on the DSQSRUN parameter.

Follow these rules when you specify variables for DSQSRUN:

- Put parentheses around the variable parameter list, as shown in the examples in this section.
- Precede the variable name with an ampersand, and make sure the string is in a *variable__name=value* format.
- Ensure the combined total of characters for the procedure name and the variable parameter list is 98 characters or less.
- Separate the variable parameter specifications using a single comma, one or more blanks, or a combination of a comma and blanks.

When you specify the name of an initial procedure, QMF issues a RUN PROC command that runs the procedure. When you use variables in your procedure, the values you supply for these variables must conform to the syntax used for passing variables on a RUN command. For information about this syntax, see the *QMF Reference* manual.

For example, suppose you frequently need two pieces of information about employees in your organization. One piece of information is the name of the

employee, and the other varies. You could define a query that includes NAME and uses a variable for the other column. Figure 54 shows an example query and procedure. The figure also shows how to pass a value for the variable when you enter the DSQSRUN parameter, and shows the RUN PROC command that QMF issues.

```
Query (named JONES.QUERY2)
        SELECT NAME, &COL FROM Q.STAFF
Procedure (named JONES.PROC2)
        RUN QUERY JONES.QUERY2 (&&COL=&COL
DSQSRUN parameter
        QMFn I=JONES.PROC2(&COL=YEARS)
Resulting RUN command
        RUN PROC JONES.PROC2 (&COL=YEARS)
```

Figure 54. Passing a QMF column name using DSQSRUN

Figure 55 shows a similar example, but instead of passing one column name to the procedure, it allows you to pass several, which return the employee's name, the department, and the employee's salary.

```
Query (named JONES.QUERY3)
        SELECT &COLS FROM Q.STAFF
Procedure (named JONES.PROC3)
        RUN QUERY JONES.QUERY3 (&&COLS=&COLS
DSQSRUN parameter
        QMFn I=JONES.PROC3(&COLS=((DEPT,NAME, SALARY))
Resulting RUN command
        RUN PROC JONES.PROC3(&COLS=((DEPT,NAME,SALARY)))
```

Figure 55. Passing several QMF column names using DSQSRUN

The next four examples show how to pass information you normally supply after the WHERE keyword in a query. (See the *QMF Reference* manual for more information about the WHERE keyword.)

These examples contain character strings. Special syntax is required due to the way QMF evaluates the values when it processes the RUN PROC command. Special characters (comma, blank, parentheses, quotes, apostrophe or single quote, and equal sign) can also be included in the string as shown.

For example, if you need to know the names and employee numbers of all the managers in your organization, you could run a query like the one in

Figure 56. When you pass the character string MGR on the DSQSRUN parameter, be sure to enclose the value in single quotes.

```
Query (named JONES.QUERY4)
        SELECT JOB, NAME, ID FROM Q.STAFF WHERE JOB=&JOB
Procedure (named JONES.PROC4)
        RUN QUERY JONES.QUERY4 (&&JOB=&JOB
DSQSRUN parameter
        QMFn I=JONES.PROC4(&JOB='MGR')
Resulting RUN command
        RUN PROC JONES.PROC4 (&JOB='MGR')
```

Figure 56. Passing a string within single quotes using DSQSRUN

Figure 57 shows how to pass variable values that contain commas. Enclose the value SAN JOSE, CA in single quotes because it contains a comma.

```
Query (named JONES.QUERY5)
        SELECT * FROM Q.APPLICANT WHERE ADDRESS+&CITY
Procedure (named JONES.PROC5)
        RUN QUERY JONES.QUERY5 (&&CITY=&CITY
DSQSRUN parameter
        QMFn I=JONES.PROC5(&CITY='SAN JOSE,CA')
Resulting RUN command
        RUN PROC JONES.PROC5 (&CITY='SAN JOSE,CA')
```

Figure 57. Passing a comma within a string using DSQSRUN

Figure 58 shows how to pass variable values that contain single quotes (for example, an apostrophe in a name). When you pass the value on the DSQSRUN parameter, be sure to enclose the value in single quotes and use two single quotes for the apostrophe instead of one.

```
Query (named JONES.QUERY6)
        SELECT * FROM Q.STAFF WHERE NAME=&NAME
Procedure (named JONES.PROC6)
        RUN QUERY JONES.QUERY6 (&&NAME=&NAME
DSQSRUN parameter
        QMFn I=JONES.PROC6(&NAME='O''BRIEN')
Resulting RUN command
        RUN PROC JONES.PROC6 (&NAME='O''BRIEN')
```

Figure 58. Passing an apostrophe as part of a string using DSQSRUN

Figure 59 shows how to pass values for variables in two different parts of the query.

```
Query (JONES.QUERY7)
        SELECT * FROM Q.STAFF WHERE DEPT IN &DEPT AND JOB=&JOB
Procedure (named JONES.QUERY7)
        RUN JONES.QUERY7 (&&DEPT=&V1 &&JOB=&V2
DSQSRUN parameter
        QMFn I=JONES.PROC7(&V1=(((10,38))) &V2='MGR')
Resulting RUN command
        RUN PROC JONES.PROC7(&V1=(((10,38))) &V2='MGR')
```

Figure 59. Passing multiple variable parameters and values using DSQSRUN

DSQSDBCS (Setting printing for double-byte character set data)

Parameter name
DSQSDBCS

Short form
K

Valid values
YES or NO

Default
NO

If you use the Uppercase or Japanese NLF, you might need to print double-byte character set (DBCS) data. You can set the DSQSDBCS program parameter to YES to print DBCS data from non-DBCS terminals.

For example, suppose a user with an IBM 3279 display terminal needs to print a table (DBCSTABLE) whose nonnumeric columns contain DBCS data. The following statement starts the Uppercase NLF from a cleared CICS screen and allows the user to print DBCSTABLE using a command such as `PRINT DBCSTABLE (PRINTER=DBCSPRT`.

```
QMFU K=YES
```

For more information on how to establish a GDDM nickname for the DBCSPRT printer, see Chapter 26, “Enabling Users to Print Objects”, on page 423.

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 39. Program parameters

Long Form	Short Form	Environment	Description
DSQSBSTG	B	TSO,CICS, CMS	Maximum storage for reports
DSQSDBCS	K	TSO,CICS, CMS	DBCS support of non-DBCS device
DSQSDBNM	D	TSO,CICS, CMS	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, CMS	Trace — ALL or NONE
DSQSIROW	F	TSO,CICS, CMS	Rows fetched from the database
DSQSMODE	M	TSO,CICS, CMS	Interactive or batch mode
DSQSPILL	L	TSO,CICS, CMS	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, CMS	Amount of reserved storage
DSQSRUN	I	TSO,CICS, CMS	Name of QMF procedure to run
DSQSSPQN	—	CICS	Name of QMF spill file
DSQSSUBS	S	TSO, CICS	Name of DB2 subsystem
DSQSUSER	UID	VSE/CICS	Authorization ID and password for DB2's database connection

Chapter 23. 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.

All sections in this chapter apply to OS/390, VM, and VSE except for the last subject, "Importing the default system initialization procedure".

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.

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 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 OS/390.

```
--
--  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 60. 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 61. 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 62. 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 63. 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 692 for more information on QMF trace options.

Importing the default system initialization procedure on OS/390

On OS/390 a default QMF system initialization procedure is shipped. The procedure is called DSQ0BINI. It can be found in QMF720.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 'QMF720.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.

Importing the default system initialization procedure on VM

A default QMF system initialization procedure called DSQ0BINI is shipped on the QMF distribution disk for VM as DSQ0BINI PROC.

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 DSQ0BINI PROC *
SAVE PROC AS Q.SYSTEM_INI (COM='DEFAULT QMF SYSTEM PROCEDURE' SHARE=YES
```

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.

Importing the default system initialization procedure on VSE

See "Installing Q.SYSTEM_INI" on page 307. You must create and save the procedure as Q.SYSTEM_INI.

Chapter 24. QMF Installation User Exit (DSQUOPTS)

DSQUOPTS, a new QMF installation user exit for QMF Version 7.2, 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.

OS/390

For OS/390, the DSQUOPTS assembler source resides in member DSQUOPTS in the QMF720.SDSQUSRE dataset. 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 QMF720.SDSQSAPE dataset. Note that the modified DSQUOPTS load module will be placed in the QMF exit library QMF720.SDSQEXIT. Remember to have the exit library properly allocated to pick up the modified exit. A default version of DSQUOPTS is shipped in the QMF720.SDSQLOAD dataset.

VM

For VM, the DSQUOPTS assembler source resides on the production disk with a filetype of ASSEMBLE. See the DSQUOPTS prolog for details on how to specify override values. DSQUOPTS requires the availability of the DSQUOPTM macro that resides in the DSQUSERE MACLIB. Follow these steps to create a modified version of DSQUOPTS:

1. Copy the DSQUOPTS ASSEMBLE source from the production disk.
2. Modify the DSQUOPTS ASSEMBLE source per instructions from the source prolog.
3. Make the DSQUSERE MACLIB available by issuing the GLOBAL MACLIB DSQUSERE command.

4. Assemble the DSQUOPTS source, HLASM DSQUOPTS.
5. Load the text file LOAD DSQUOPTS.
6. Generate the module GENMOD DSQUOPTS (AMODE 31 RMODE ANY).

VSE

Modification of the DSQUOPTS user exit is not available for VSE.

Chapter 25. 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 to QMF on OS/390

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 the *QMF Reference* manual.)
- Default views for the database object list, explained in "Customizing a user's database object list" on page 366.

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 317 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 the *Using QMF* manual) to add new user profiles to the Q.PROFILES table. Figure 64 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 64 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 64. 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 64 shows only a subset of all possible profile values. Use "Reading the Q.PROFILES table" on page 317 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 the *QMF Reference* manual .

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 65 shows SQL statements that delete the rows. You can also use the Table Editor, as explained in *Using 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 65. 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 40 on page 318 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 xiv.)

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 40. Structure of the Q.PROFILES table

Column name	Data type and length	Nulls allowed	Function and possible values for OS/390
CREATOR	CHAR (8)	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 the *QMF Reference* manual 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 the *QMF Reference* manual 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 40. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for OS/390
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 TABLESPACE name. Operation is not affected. DataJoiner does not utilize tablespaces and the value for the SPACE option is ignored in a 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 40. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for OS/390
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.
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 xiv shows the translated names you need to use in this column.
PFKEYS	VARCHAR (31)	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	VARCHAR (31)	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.
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).

Table 40. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for OS/390
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 OS/390.

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)
```

Providing the correct profile on OS/390

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, OS/390, 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 *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 the *Using QMF* manual.

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 66 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 66. 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 383.
- For instructions on database tables and views, see "Maintaining tables and views using DB2 tables" on page 398.

Use a query similar to the one shown in Figure 67 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 67. 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 OS/390

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. The *DB2 UDB for OS/390 SQL Reference* manual explains the DROP statement. The *QMF Reference* manual explains the ERASE command.

Establishing QMF support on VM

The role of Q.AUTHID: QMF installation automatically grants DBA 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 the *QMF Reference* manual.)
- Default views for the database object list, explained in "Customizing a user's database object list" on page 366.

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

Ensuring that users have access to CMS

Provide a new user with a VM logon ID. Set up the new users as you would a DB2 for VM user virtual machine. See DB2 Server for *VM Database Administration* for more information.

To communicate with DB2 for VM, a new QMF user logging on for the first time must issue this command (assuming the user is linked to the DB2 for VM production disk):

```
SQLINIT DBNAME(dbname)
```

where *dbname* is the name of the database that is being used for QMF. That command loads two required modules to the user's A-disk. As long as those modules remain, and as long as the user wants to use the same database, the command does not need to be reissued. Log on with the new user ID, and give the SOLINIT command for the new user.

If your users need to connect to DB2 for VM explicitly, grant them DB2 for VM CONNECT authority:

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

The QMF CONNECT command enables an individual to access DB2 for VM using an established CONNECT ID (DB2 for VM user ID), or to connect to a different database during a QMF session. This command is useful for running jobs in batch mode.

After a user has received CONNECT authority (has been assigned a DB2 for VM user ID), the user can access DB2 for VM through the QMF CONNECT command:

```
CONNECT userid(PASSWORD=password
```

userid Any user ID conforming to the VM logon ID syntax rules is acceptable. However, only those IDs that have been granted access to DB2 for VM can be used in the CONNECT command. The ID can be embedded in double quotation marks.

DB2 for VM password The DB2 for VM password:

- Must have no more than eight characters.
- Can be embedded in single or double quotation marks. A single quotation mark embedded within single quotation marks is removed.
- Must contain no blanks (except for trailing blanks).

In order for a user to use the CONNECT command, the user ID and password must both be in SYSTEM.SYSUSERAUTH. The password does not need to be the same as the one associated with the VM logon ID.

In order for a user to use the CONNECT command, the user ID and password must both be in SYSTEM.SYSUSERAUTH. The password does not need to be the same as the one associated with the VM logon ID.

As a result of a QMF CONNECT command, the QMF profile resets to the password associated with the new DB2 for VM user ID or to the SYSTEM row default if that DB2 for VM user ID is not represented in Q.PROFILES. For more information about CONNECT authority, see the*DB2 server for VM Database Administration* manual.

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 "Reading the Q.PROFILES table" on page 327. 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 user, as shown in "Adding a new user profile to the Q.PROFILES table in CMS" on page 326. Set the CREATOR column of Q.PROFILES to the primary authorization ID of the user and customize other column values according to individual needs.

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 in CMS

You can use SQL INSERT queries or the QMF Table Editor (described in the *Using QMF* manual) to add new user profiles to the Q.PROFILES table. Figure 68 shows sample SQL that creates unique profiles 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.

```
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', 'CMS')
```

```
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'
    'CMS')
```

Figure 68. Creating a user profile in CMS

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. You can also use the Table Editor, as explained in *Using 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 69. 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 41 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 xiv.)

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 41. Structure of the Q.PROFILES table

Column name	Data type and length	Nulls allowed	Function and possible values for VM
CREATOR	CHAR (8)	No	**Function:** Specifies the authorization ID (the user) who owns the profile. **Values:** SYSTEM (default), primary or SQL authorization ID, 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 the *QMF Reference* manual for descriptions of these values. CASE might have a different default for NLF users.

Table 41. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for VM
DECOPT	CHAR (18)	Yes	**Function:**Specifies what separators QMF puts in numeric report columns. **Values:**PERIOD (default), COMMA, and FRENCH. See the *QMF Reference* manual 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.
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 dbspace that holds tables created using SAVE DATA and IMPORT commands. **Values:** Any valid dbspace name.

Table 41. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for VM
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.
PRINTER	CHAR (8)	Yes	**Function:** Controls where printer output is routed. **Values:** Use a null (default) or blank value to route print output to a file or printer associated with the DSQPRINT FILEDEF. Use a GDDM nickname to direct output to a GDDM-defined printer.
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 xiv shows the translated names you need to use in this column.
PFKEYS	VARCHAR (31)	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	VARCHAR (31)	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 41. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for VM
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 null or CMS. If profiles are stored in DB2 for VM, but are being accessed from a DB2 application requestor, the value used for ENVIRONMENT can be TSO or CICS.

Providing the correct profile for VM

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. CREATOR=*userid*, ENVIRONMENT=NULL
3. CREATOR=SYSTEM, ENVIRONMENT=*current operating environment*
4. CREATOR=SYSTEM, ENVIRONMENT=NULL

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

Current operating environment is CMS when QMF is being started from CMS.

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 *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 the *Using QMF* manual.

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 70 on page 332 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 70. 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 383.
- For instructions on database tables and views, see “Maintaining tables and views using DB2 tables” on page 398.

Use a query similar to the one shown in Figure 71 on page 333 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 71. 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 VM

If the user whose profile you deleted had a private dbspace, use the SQL DROP DBSPACE 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. The *DB2 for VSE & VM SQL Reference* manual explains the DROP statement. The *QMF Reference* manual explains the ERASE command.

When you delete a user profile, all SQL privileges the user had on objects are deleted, as well as all privileges that the user granted to other users. To ensure other users will not be affected, query the SYSTEM.SYSTABAUTH table to see what SQL privileges have been granted to the user. Query the SYSTEM.SYSUSERAUTH table to see what DB2 authorities have been granted.

Establishing QMF support on VSE

The role of Q.AUTHID: QMF installation automatically grants DBA 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 the *QMF Reference* manual.)
- Default views for the database object list, explained in "Customizing a user's database object list" on page 366.

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 "Reading the Q.PROFILES table" on page 335. 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 user, as shown in "Adding a new user profile to the Q.PROFILES table in CICS/VSE". Set the CREATOR column of Q.PROFILES to the primary authorization ID of the user and customize other column values according to individual needs.

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 in CICS/VSE

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

```
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', 'CICSVSE')
```

```
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'
    'CICSVSE')
```

Figure 72. Creating a user profile in CICSVSE

Ensuring that users have access to CICS

Before setting up user access to QMF, make sure the user is known to CICS. Define a three-character CICS terminal operator ID by defining a VSE user ID. Map the VSE user ID to a CICS terminal operator ID, and redefine the CICS ID in the the default sign-on table (SNT) that is shipped with VSE.

If your users need to connect to DB2 explicitly, grant them DB2 CONNECT authority:

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

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 73 shows SQL statements that delete the rows. You can also use the Table Editor, as explained in *Using 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 73. 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 42 on page 336 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 xiv.)

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 42. Structure of the Q.PROFILES table

Column name	Data type and length	Nulls allowed	Function and possible values for VSE
CREATOR	CHAR (8)	No	**Function:** Specifies the authorization ID (the user) who owns the profile. **Values:** SYSTEM (default), primary or SQL authorization ID. 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 the *QMF Reference* manual 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 the *QMF Reference* manual 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.
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.

Table 42. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for VSE
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 dbspace that holds tables created using SAVE DATA and IMPORT commands. **Values:** Any valid dbspace 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. Only the values ALL and NONE are translated in NLFs.
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.
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 xiv shows the translated names you need to use in this column.

Table 42. Structure of the Q.PROFILES table (continued)

Column name	Data type and length	Nulls allowed	Function and possible values for VSE
PFKEYS	VARCHAR (31)	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	VARCHAR (31)	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.
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 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 CICSVSE if you access the profile through CICS/VSE. If profiles are stored in DB2 for VSE, but are being accessed from a DB2 application requestor, the value used for ENVIRONMENT can be TSO, CMS, or CICS.

Providing the correct profile for VSE

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=*auth ID*, ENVIRONMENT=*current operating environment*
2. If running in CICS, CREATOR=*auth ID*, ENVIRONMENT=CICS
3. CREATOR= *auth ID*, ENVIRONMENT=NULL
4. CREATOR=SYSTEM, ENVIRONMENT=*current operating environment*
5. If running in CICS, CREATOR=SYSTEM, ENVIRONMENT=CICS
6. CREATOR=SYSTEM, ENVIRONMENT=NULL

auth ID is the DB2 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 CICSVSE if the profiles are stored in VSE DB2 and are being accessed through CICS/VSE.

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.

Storing profiles in VM DB2 in a guest-sharing environment

If you store QMF VSE profiles in a DB2 VM database, add the values CICSVSE to the ENVIRONMENT column of the user's QMF VM profile to ensure that your users can access QMF. Figure 74 shows how a site using DB2 guest sharing might use QMF VSE to access profiles and other objects stored in DB2 VM.

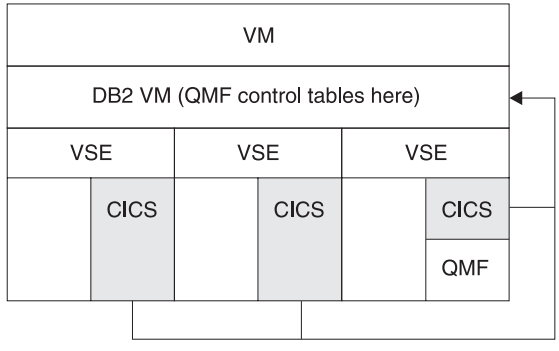


Figure 74. Possible guest sharing scenario for profiles

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 *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 the *Using QMF* manual.

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 75 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 75. 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 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 383.
- For instructions on database tables and views, see "Maintaining tables and views using DB2 tables" on page 398.

Use a query similar to the one shown in Figure 76 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 76. 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 VSE

If the user whose profile you deleted had a private dbspace, use the SQL DROP DBSPACE 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. The *DB2 for VSE & VM SQL Reference* manual explains the DROP statement. The *QMF Reference* manual explains the ERASE command.

When you delete a user profile, all SQL privileges the user had on objects are deleted, as well as all privileges that the user granted to other users. To ensure other users will not be affected, query the SYSTEM.SYSTABAUTH table to see what SQL privileges have been granted to the user. Query the SYSTEM.SYSUSERAUTH table to see what DB2 authorities have been granted.

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 you are the owner of the table, use the statement in Figure 77to 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 77. 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 the *QMF Reference* manual. 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 in Figure 78 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 78. Granting an SQL privilege to all QMF users

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

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 79. 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, as discussed in "Granting and revoking SQL privileges" on page 363. "SQL privileges required to access objects" on page 361 explains privileges required to use specific QMF commands or functions on objects.
- You can use the SHARE parameter of the QMF SAVE command to control access to queries, forms, and procedures, as discussed in "Sharing QMF objects with other users" on page 365.

Controlling access on OS/390

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 OS/390 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 353.

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 OS/390 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 OS/390, 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 for OS/390 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 appropriate *DB2 UDB SQL Reference* manual.

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 OS/390: 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 OS/390: 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 OS/390 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 OS/390 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 TABLESPACE &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 OS/390: 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 OS/390: 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 the *DB2 UDB for OS390 Administration Guide* Also, see the description of the REVOKE command in the *DB2 UDB for OS390 SQL Reference* manual.

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 43, 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 43. 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 the appropriate *DB2 SQL Reference* manual.

SQL privileges required for prompted and QBE queries: Using Table 44, 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 44. 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 QMF* .

SQL privileges required for the table editor: Using Table 45, 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 45. 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 80. 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 the *QMF Reference* manual. 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 81. Granting an SQL privilege to all QMF users

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

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 82. Sharing a QMF object

The default is defined by the global variable DSQEC_SHARE. See the *QMF Reference* manual 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 83. Changing the shared status of a QMF object

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

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 QMF Version 7.2 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 *QMF Reference*.

Controlling access on VM

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 might need to control users' access to certain objects.

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 46, locate the QMF command your users need to use and grant them the required SQL privilege on the table or view they're working with. See "Granting and revoking SQL privileges" on page 342for examples of SQL GRANT statements.

Table 46. 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

Table 46. QMF commands and their SQL equivalents (continued)

This QMF command:	Requires this SQL privilege on objects referenced by the command:
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 the appropriate *DB2 SQL Reference* manual.

SQL privileges required for prompted and QBE queries: Using Table 47, 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 47. 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 QMF* .

SQL privileges required for the table editor: Using Table 48, 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 48. 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 below 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 84. 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 the *QMF Reference* manual. 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 85. Granting an SQL privilege to all QMF users

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

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 86. Sharing a QMF object

The default is defined by the global variable DSQEC_SHARE. See the *QMF Reference* manual 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 87. Changing the shared status of a QMF object

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

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 QMF Version 7.2 the use of the value '0' is only effective with the database servers DB2 for VM Version 5 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 *QMF Reference*.

Controlling access on VSE

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 might need to control users' access to certain objects.

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 49, locate the QMF command your users need to use and grant them the required SQL privilege on the table or view they're working with. See "Granting and revoking SQL privileges" on page 342 for examples of SQL GRANT statements.

Table 49. 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

Table 49. QMF commands and their SQL equivalents (continued)

This QMF command:	Requires this SQL privilege on objects referenced by the command:
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 the appropriate *DB2 SQL Reference* manual.

SQL privileges required for prompted and QBE queries: Using Table 50, 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 50. 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 QMF* .

SQL privileges required for the table editor: Using Table 51 on page 363, 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 51. 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*.

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 you are the owner of the table, use the statement in tFigure 88o 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 88. 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 the *QMF Reference* manual. 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 89. Granting an SQL privilege to all QMF users

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

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 90. 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.

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=NO
```

Figure 91. Sharing a QMF object on VSE

The default is defined by the global variable DSQEC_SHARE. See the *QMF Reference* manual 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 92. Changing the shared status of a QMF object

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

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 QMF Version 7.2 the use of the value '0' is only effective with DB2 for VSE Version 5 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 *QMF Reference*.

Customizing a user's database object list

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 on OS/390

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 for OS/390 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 for OS/390, and workstation or iSeries database servers. Figure 93 shows the view provided for DB2 for OS/390.

```
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 93. Default view that provides a list of tables for the LIST command (OS/390)

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 OS/390.

```
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 94. Default view that provides a list of tables for the LIST command (OS/390)

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 for OS/390, 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 95. Default view that provides a list of aliases for the LIST command (OS/390)

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 96. Default view that provides column information for the DESCRIBE command (OS/390)

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 97 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 the *QMF Reference* manual 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 97. Customizing your object lists using global variables (OS/390)

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 QMF*. For a list of global variables and information on using them in applications, see the *Developing QMF Applications* manual .

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 729.

Using the default object lists on VM and VSE

QMF provides the following default views and automatically assigns them to the user Q during installation into DB2 databases:

Q.DSQEC__TABS__SQL
Q.DSQEC__COLS__SQL

QMF supplies a variation of the following views when QMF is installed into a DB2 workstation server or DB2 iSeries:

Q.DSQEC__TABS__LDB2
Q.DSQEC__ALIASES
Q.DSQEC__COLS__LDB2

The view Q.DSQEC_TABS_SQL selects only those database tables the user is authorized to see. Figure 98 shows the view provided for DB2. To override the default view Q.DSQEC_TABS_SQL, issue a command like this:

```
CREATE VIEW Q.DSQEC_TABS__SQL
    (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,' ',' ',' ',
    TLABEL,' ',' ',' '
    FROM SYSTEM.SYSCATALOG, SYSTEM.SYSTABAUTH
    WHERE CREATOR = TCREATOR AND TNAME=TTNAME AND GRANTEETYPE = ' ' AND
         GRANTEE IN (USER,'PUBLIC');
COMMENT ON TABLE Q.DSQEC_TABS_SQL IS
                  'QMF VIEW FOR DB2 TABLES/VIEWS LIST';
GRANT SELECT ON Q.DSQEC_TABS_SQL TO PUBLIC;
```

Figure 98. Default view that provides a list of tables for the LIST command

The view Q.DSQEC_COLS_SQL selects only the column information a user is authorized to see in DB2 database servers. Figure 99 shows the view provided:

```
CREATE VIEW Q.DSQEC__COLS__SQL
    (OWNER, TNAME, CNAME, REMARKS,LABEL)
  AS SELECT
    CREATOR, TBNAME, CNAME, REMARKS, CLABEL
  FROM SYSTEM.SYSCOLUMNS, SYSTEM.SYSTABAUTH
    WHERE TCREATOR = CREATOR AND TTNAME = BNAME AND GRANTEETYPE = ' '
      AND GRANTEE IN (USER,'PUBLIC')
```

Figure 99. Default view that provides column information for the DESCRIBE command

To override the default view Q.DSQEC_COLS_SQL, issue the command:

```
SET GLOBAL (DSQEC_COLS_SQL = userid_hour_local_sql_columns
```

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 SYSTEM.SYSCATALOG and SYSTEM.SYSCOLUMNS tables. 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 below 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 the *QMF Reference* manual 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 SYSTEM.SYSCATALOG.A
   WHERE TNAME IN (SELECT TTNAME
                FROM SYSTEM.SYSTABAUTH
                 WHERE TCREATOR = A.CREATOR
                   AND GRANTEETYPE = ' &'
                   AND GRANTEE IN (USER, 'PUBLIC'))
```

Figure 100. 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 SYSTEM.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 SYSTEM.SYSCATALOG for table list or SYSTEM.SYSCOLUMNS for column list. Then the administrators can name this view in the DSQEC__COLS__SQL or DSQEC__TABS__SQL global variables and access descriptive information for any columns in the database.

Follow these rules if you are 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__SQL and DSQEC__COLS__SQL 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 QMF*. For a list of global variables and information on using them in applications, see the *Developing QMF Applications* manual .

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 729.

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 the *QMF Reference* manual 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 the *QMF Reference* manual 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 OS/390

Table 52. 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 Administration Guide* to help you decide on assigning authorities to create table spaces or dbspaces.	**Step 1**	Use the *DB2 UDB for OS390 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.
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 the *Using QMF* manual 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 OS390 SQL Reference* manual 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 OS/390 LOAD Utility, QMF IMPORT commands (for transferring small tables), or other methods. The *DB2 UDB for OS390 Uitility Guide and Reference* manual explains how to use the LOAD Utility. The *Using QMF* manual 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 the *QMF Reference* manual for more information on these commands.

Table 52. Creating tables in the database (continued)

If you are creating tables for your users:	If users are creating tables themselves:
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 the *DB2 UDB for OS390 SQL Reference* manual.

Choosing and assigning a table space for the user (OS/390)

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 the *DB2 UDB for OS390 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 the *DB2 UDB for OS390 SQL Reference* manual.

For information on the table spaces, see the *DB2 UDB for OS390 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 UDB for OS390 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 OS390 Administration Guide* .

Granting a user DB2 CREATETAB authority (OS/390)

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 101, where *userid1*, *userid2*, and *userid3* represent SQL authorization IDs.

```
GRANT CREATETAB on database DBASEA TO userid1, userid2, userid3, ...
```

Figure 101. 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 *DB2 UDB for OS390 Administration Guide* for more information on creating a table space and a discussion of DB2 authority levels.

Creating tables on VM and VSE

Table 53. Creating tables in the database

If you are creating tables for your users:	If users are creating tables themselves:
Step 1 Acquire a dbspace and define it to DB2 for VM before its first use. Use the appropriate *DB2 Administration Guide* to help you decide on assigning authorities to create table spaces or dbspaces.	**Step 1** Acquire a dbspace and define it to DB2 before its first use. Use the *DB2 Server for VSE & VM Administration Guide* to help you decide on a private or public dbspace.
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 the *Using QMF* manual for examples of creating tables.	**Step 2** Assign the dbspace 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.	**Step 3** Grant DB2 RESOURCE authority to users creating their own tables in public dbspaces, or acquire a private dbspace for the user. Users automatically have all SQL privileges on tables they create.
Step 4 Fill the tables with data. Use the DB2 DBS utility, QMF IMPORT commands (for transferring small tables), or other methods. The *Using QMF* manual 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 the *QMF Reference* manual for more information on these commands.
Step 5 Grant SQL privileges for the tables to users who need them.	**Step 5** Grant 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.

Choosing and acquiring a dbspace for the user

A dbspace can be either private or public. Any QMF user with DB2 RESOURCE authority can create tables in a public dbspace. If the dbspace is private, only the assignee is allowed to create tables in it. For additional guidance on types of dbspaces, see the *DB2 Server for VSE & VM Database Administration* manual.

Using the SQL ACQUIRE statement: After you decide whether a public or private dbspace best suits your needs, acquire the dbspace using a statement

similar to the one in below. You can enter this statement from the QMF SQL query panel, then press the Run function key to run the query.

```
ACQUIRE PUBLIC DBSPACE NAMED dbspacename
(PAGES=1024)
```

Figure 102. Acquiring a dbspace

Substitute PRIVATE for PUBLIC in the statement if you are acquiring a private dbspace and be sure to qualify dbspacename with the SQL authorization ID of the user for whom you are acquiring the dbspace.

Sizing a dbspace: The size of the dbspace in an acquire statement is given in *pages*, where one page is 4,096 bytes. If you do not specify a page size, a default value of 128 pages is assumed. Estimate the size you need by estimating the size of the tables the dbspace must hold, as though the tables are reports and you are estimating the size of a spill file to hold them. "Estimating the Space Required for a Spill File" on page 76 shows an algorithm for estimating the size of a spill file.

Whatever size you choose, first search the DB2 storage pools for an existing dbspace close to the size you need. If no dbspace of convenient size already exists, use the ADD dbspace statement to create one. Instructions for adding dbspaces are provided in the *DB2 Server for VSE & VM System Administration* manual.

Granting a user DB2 RESOURCE authority

You need to grant DB2 RESOURCE authority to any user who needs to create tables in a public dbspace. To grant a user RESOURCE authority, issue the SQL statement shown below where userid1, userid2 and userid3 represent SQL authorization IDs.

```
GRANT RESOURCE TO userid1, userid2, userid3...
```

A user with RESOURCE authority can acquire a private dbspace for his or her own use, and create tables in a public dbspace in addition to those created in a private dbspace.

If you want to allow a user to create tables, but need to maintain control over how much resources are used, acquire a private dbspace for the user rather than granting RESOURCE authority. That way, you can control the size of the dbspace and the amount of resources used. See the *DB2 Server for VSE & VM Database Administration* manual for more information on acquiring a dbspace and a discussion of DB2 authority levels.

Enabling users to confirm table changes before they are made: Using the QMF Table Editor, a user can add, delete, or update information in a database

table. If the value of the CONFIRM field of a user's QMF profile is YES, QMF displays a panel before making database changes. This panel asks users if they are sure they want to change the database.

To enable users to confirm their database changes, first make sure the dbspace you chose for the user is recoverable. Because changes to DB2 tables stored in nonrecoverable table spaces or dbspaces cannot be rolled back or canceled, answering NO on the Table Editor confirmation prompt panel for database changes will not prevent the changes to the table from taking place.

As end users become more comfortable changing data in the database, they may not need QMF to display these confirmation panels. You can use the following global variables to disable the panels for specific categories of actions allowed by the Table Editor:

- DSQCP_TEADD for the ADD category
- DSQCP-TECHG for the CHANGE category
- DSQCP_TEDEL for the DELETE category
- DSQC[_TEEND for the END/CANCEL category
- DSQCP_TEMOD for the MODIFY category

The Table Editor loads values for these variables when it is initialized. The possible values for each variable are:

- **0** Disables the confirmation panel for the category
- **1** Enables the confirmation panel for the category
- **2** (the default) Either disables or enables the panel for the category depending on how the SAVE keyword of the EDIT command is set:
  - When SAVE=IMMEDIATE, the confirmation panel displays
  - When SAVE=END, the confirmation panel displays for the DELETE, MODIFY, and END/CANCEL categories, but does not display for the ADD and CHANGE categories.

For more information about functions provided by the QMF Table Editor, see the *Using QMF* manual.

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 QMF720.DSQCHART. 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 QMF720.DSQCHART.

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 OS/390

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.

Supporting a chart on VM

On VM, the IBM-supplied chart forms are supplied on the QMF production disk. When the user saves a chart form, it is saved on the user's A-disk. Charts on a user's A-disk are used before charts on the QMF production disk. 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. On VSE, user-defined chart objects are saved in the GDDM file ADMF. This file is defined during GDDM for CICS.

Supporting a chart on VSE

If you install GDDM-PGF after installing 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 3.2, you need to install GDDM-PGR 2.1.3. If you use GDDM 2.3, you need to install GDDM-PGF 2.1.1.

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 since these formats were defined for you during QMF installation.

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" on page 384

- The Q.OBJECT__DATA table, which is described in "Reading the Q.OBJECT__DATA table" on page 385
- The Q.OBJECT__REMARKS table, which is described in "Reading the Q.OBJECT_REMARKS table" on page 386

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.

Workstation database server users: DB2 Common Server has an additional control table, Q.OBJECT_DATA2. QMF users must have INSERT privilege to this table. When all database activity is complete, this table should contain no records. If any records remain, they might be removed.

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 54:

Table 54. 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.
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.

Table 54. Structure of the Q.OBJECT__DIRECTORY table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
RESTRICTED	CHAR	1	No	YES if the object has not been shared (using the SHARE parameter of the QMF SAVE command); 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 55:

Table 55. Structure of the Q.OBJECT__DATA 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 55. Structure of the Q.OBJECT__DATA table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
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)	3600 (see note)	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.

Note: With DataJoiner Version 1.2.1 and DB2 for AIX, Parallel Edition Version 1.2, the data type and length for APPLDATA are VARCHAR(3600) for bit data. This is a permanent restriction for Version 1 SQL databases.

Workstation database server users: For DB2 Common Server, there is a similar table, Q.OBJECT_DATA2. This table is required for internal QMF processing of a SAVE or IMPORT command.

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 *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 56:

Table 56. Structure of the Q.OBJECT__REMARKS table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	CHAR	8	No	Shows the authorization ID of the user who created the object

Table 56. Structure of the Q.OBJECT__REMARKS table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
NAME	VARCHAR	18	No	Shows the name of the object.
TYPE	CHAR	8	No	Shows the type of the object: FORM, PROC, or QUERY.
REMARKS	VARCHAR	254	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 103.

```
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 103. Listing queries, 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 104. 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 105 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 105. Transferring QMF objects to another user

In the queries shown in Figure 105, `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 104 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 106 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 106. 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 OS/390 data sets

If a user has QMF objects that have been exported to OS/390 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 57. 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 58. Node groups for control tables that store QMF objects using a DB2 Parallel Edition V1R2 database or DB2 UDB

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 58. Node groups for control tables that store QMF objects using a DB2 Parallel Edition V1R2 database or DB2 UDB (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 the *DB2 UDB for OS390 Uitility Guide and Reference* manual.

Note: QMF Version 7.2 creates DB2 managed table space data sets if QMF was not previously installed.

Enlarging the dbspace for the QMF object control tables on VM

Periodically, QMF objects might become too large for the dbspace that contains the QMF object control tables Q.OBJECT_DIRECTORY, Q.OBJECT_DATA, AND Q.OBJECT_REMARKS.

Use the DB2 for VM DBS utility to enlarge the dbspace for the QMF object control tables:

1. Archive the database so that a backup copy is available for recovery if needed.
2. Unload the dbspace to a CMS sequential file using the UNLOAD dbspace command of the DBS utility.

   Table 59 shows the dbspace names and default sizes for the QMF object control tables. Dbspace names for other QMF control tables are shown in "Appendix D. QMF Control Tables and Dbspaces Used by QMF" on page 325.

   All dbspaces for the QMF control tables are public. The sizes are given in pages, where each page is one 4,096-byte block.

Table 59. Dbspaces for control tables that store QMF objects

Dbspace name	Contents	Default size
DSQTSCT1	Q.OBJECT_DIRECTORY table	256 pages
DSQTSCT2	Q.OBJECT_REMARKS table	256 pages
DSQTSCT3	Q.OBJECT_DATA table	5120 pages

3. Drop the dbspace using the DBS utility or ISQL.
4. Acquire a larger public space for the dbspace using either the DBS utility or ISQL. For example:

```
ACQUIRE PUBLIC DBSPACE NAMED PUBLIC.DSQxxxxx
   (PAGES=xxx, PCTFREE=25, LOCK=ROW)
```

5. Use the DBS utility to reload the QMF object control tables into the new dbspace with the file you specified when you unloaded the tables as the new input file. Use the NEW keyword for the RELOAD dbspace command.
6. Recreate indexes for the reloaded tables using the DBS utility or ISQL. Make sure that:
   - The indexes are unique.
   - The index name for the Q.OBJECT_DIRECTORY table is OBJECT_DIRECTORYX and is keyed on the OWNER and NAME columns.
   - The index name for the Q.OBJECT_DATA table is OBJECT_OBJDATAX and is keyed on the OWNER, NAME, and SEQ columns.

- The index name for the Q.OBJECT_REMARKS table is OBJECT_REMARKSX and is keyed on the OWNER and NAME columns.

7. Recreate views if the dbspaces for Q.OBJECT_DIRECTORY or Q.OBJECT_REMARKS were dropped. For example: To provide access to this view to all QMF users, grant SELECT authority.

```
CREATE VIEW Q.DSQEC_QMFOBJS
   (OWNER, TNAME, TYPE, SUBTYPE, MODEL, RESTRICTED, REMARKS, LABEL,
    LOCATION, OWNER_AT_LOCATION, NAME_AT_LOCATION)
AS SELECT
   A.OWNER, A.NAME, A.TYPE, SUBTYPE, MODEL, RESTRICTED,
     REMARKS, ' ', ' ', ' '
   FROM Q.OBJECT_DIRECTORY A, Q.OBJECT_REMARKS B
   WHERE A.OWNER = B.OWNER AND A.NAME = B.NAME
   AND (A.OWNER = USER OR RESTRICTED = 'N')
```

   to PUBLIC:

```
GRANT SELECT ON Q.DSQEC_QMFOBJS TO PUBLIC
```

8. Alter the dbspace to allow the free space on occupied pages to e used. For example:

```
ALTER DBSPACE PUBLIC.DSQTSCT1 (PCTFREE=5)
```

9. If you change the QMF control tables, reload the QMF SQL packages with the install exec DSQ2PREP EXEC on VM.

For more information on enlarging dbspaces, see the appropriate *DB2 Server Database Administration Guide*. For instructions and syntax of the DBS utility and ISQL commands, see the*DB2 Server for VSE and VM Database Services Utility* manual and the *DB2 Server for VSE and VM SQL Reference* manual.

Enlarging the dbspace for the QMF object control tables on VSE

Periodically, QMF objects might become too large for the dbspace that contains the QMF object control tables Q.OBJECT_DIRECTORY, Q.OBJECT_DATA, AND Q.OBJECT_REMARKS.

Use the DB2 DBS utility to enlarge the dbspace for the QMF object control tables:

1. Archive the database so that a backup copy is available for recovery if needed.
2. Unload the dbspace using the UNLOAD dbspace command of the DBS utility.

   Table 60 on page 394 shows the dbspace names and default sizes for the QMF object control tables. Dbspace names for other QMF control tables are shown in ″Appendix D. QMF Control Tables and Dbspaces Used by QMF″ on page 325.

All dbspaces for the QMF control tables are public. The sizes are given in pages, where each page is one 4,096-byte block.

Table 60. Dbspaces for control tables that store QMF objects

Dbspace name	Contents	Default size
DSQTSCT1	Q.OBJECT_DIRECTORY table	256 pages
DSQTSCT2	Q.OBJECT_REMARKS table	256 pages
DSQTSCT3	Q.OBJECT_DATA table	5120 pages

3. Drop the dbspace using the DBS utility or ISQL.
4. Acquire a larger public space for the dbspace using either the DBS utility or ISQL. For example:

```
ACQUIRE PUBLIC DBSPACE NAMED PUBLIC.DSQxxxxx
   (PAGES=xxx, PCTFREE=25, LOCK=ROW)
```

5. Use the DBS utility to reload the QMF object control tables into the new dbspace with the file you specified when you unloaded the tables as the new input file. Use the NEW keyword for the RELOAD dbspace command.
6. Recreate indexes for the reloaded tables using the DBS utility or ISQL. Make sure that:
   - The indexes are unique.
   - The index name for the Q.OBJECT_DIRECTORY table is OBJECT_DIRECTORYX and is keyed on the OWNER and NAME columns.
   - The index name for the Q.OBJECT_DATA table is OBJECT_OBJDATAX and is keyed on the OWNER, NAME, and SEQ columns.
   - The index name for the Q.OBJECT_REMARKS table is OBJECT_REMARKSX and is keyed on the OWNER and NAME columns.
7. Recreate views if the dbspaces for Q.OBJECT_DIRECTORY or Q.OBJECT_REMARKS were dropped. For example: To provide access to this view to all QMF users, grant SELECT authority.

```
CREATE VIEW Q.DSQEC_QMFOBJS
   (OWNER, TNAME, TYPE, SUBTYPE, MODEL, RESTRICTED, REMARKS, LABEL,
    LOCATION, OWNER_AT_LOCATION, NAME_AT_LOCATION)
AS SELECT
   A.OWNER, A.NAME, A.TYPE, SUBTYPE, MODEL, RESTRICTED,
     REMARKS, ' ', ' ', ' '
   FROM Q.OBJECT_DIRECTORY A, Q.OBJECT_REMARKS B
   WHERE A.OWNER = B.OWNER AND A.NAME = B.NAME
   AND (A.OWNER = USER OR RESTRICTED = 'N')
```

to PUBLIC:

```
GRANT SELECT ON Q.DSQEC_QMFOBJS TO PUBLIC
```

8. Alter the dbspace to allow the free space on occupied pages to e used. For example:

   ```
   ALTER DBSPACE PUBLIC.DSQTSCT1 (PCTFREE=5)
   ```

9. If you change the QMF control tables, reload the QMF SQL packages with DSQ3EDBI JCLE on VSE.

For more information on enlarging dbspaces, see the appropriate *DB2 Server Database Administration Guide*. For instructions and syntax of the DBS utility and ISQL commands, see the*DB2 Server for VSE and VM Database Services Utility* manual and the *DB2 Server for VSE and VM SQL Reference* manual.

Maintaining a DB2 subsystem on OS/390

Note: Except where noted, this section provides information about DB2 for MVS/ESA.

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.

DB2 common server users: QMF table spaces are defined as system managed space (SMS).

Storage groups for DB2 managed data sets

Prior to Version 3.2, DB2 managed the space for the control table indexes and table spaces. This required the use of a storage group for each table space and index. 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 61 indicates.

Workstation database server users: Storage groups do not apply.

Table 61. 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 OS390 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 the *DB2 UDB for OS390 Uitility Guide and Reference* manual.

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 OS390 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 OS390 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.

Determining index use

QMF query performance can be affected if the QMF application plan is bound when Q.OBJECT_DATA has very few entries. Under these circumstances, the index on Q.OBJECT_DATA is not being used by the optimizer. (The optimizer is a DB2 function that determines the best ways to access a row in a table.) Instead, a table space scan is performed, affecting future performance when Q.OBJECT_DATA contains many entries. You need to rebind the plan so that the index is used.

To determine whether the index on Q.OBJECT_DATA is being used, run the following query:

```
SELECT BCREATOR, BNAME
  FROM SYSIBM.SYSPLANDEP
    WHERE DNAME='QMF720'
      AND BTYPE='I'
```

This query selects the owner (BCREATOR) and name (BNAME) of any indexes that the QMF application plan is dependent upon. Although QMF720 is the default plan name, use the name used during QMF installation.

If the result does not indicate an entry for Q.OBJECT_OBJDATAX (Q.OBJECT_DATAX, if you are migrated from QMF V2R2), do the following:
1. Run RUNSTATS on table space DSQDBCTL.DSQTSCT3.
2. Rebind the QMF application plan.

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 the *DB2 UDB for OS390 SQL Reference* manual. 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 OS/390

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 107 on page 399 returns a list of tables from DB2 OS/390 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 107. 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.*

Using DB2 for VM and VSE System tables

Anyone with DBA authority can access the DB2 tables to list, display, transfer, or delete tables and views. Transferring ownership of a table or view is not recommended.

Listing tables and views

The query in Figure 108returns a list of tables from DB2 VM with columns TABLETYPE (R indicates a table, V indicates a view), TNAME (table name), TABESPACENAME, and REMARKS.

```
SELECT TABLETYPE, TNAME, DBSPACENAME, REMARKS
   FROM SYSTEM.SYSCATALOG
   WHERE CREATOR = 'userid'
   ORDER BY TABLETYPE, TNAME
```

Figure 108. Listing DB2 tables and views owned by a particular user (VM)

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 SYSTEM.SYSCATALOG 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).

Supporting locally defined date/time formats

Note: Locally defined date/time formats are not supported in CICS.

Locally defined date/time formats on OS/390

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 OS390 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 109, 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 109. Making the edit routine available

Locally defined date/time formats on VM

QMF's support of DATE, TIME and TIMESTAMP data types makes it possible for your users to use local date/time exit routines. When planning for local date/time exits, remember that theses are DB2 for VM exits, not QMF exits. For details about how these exits are created, refer to the *DB2 Server for VM System Administration* manual.

For QMF to use a local date/time exit, the text files containing the date/time exits "ARIUXDT" and "ARIUXTM" must be placed on a minidisk that is accessible to QMF when QMF starts. If QMF is started using DCSS mode, two relocatable module files must be created from the existing exit text files "ARIUXDT" and "ARIUXTM". To create the relocatable module files, issue the following CMS commands:

```
LOAD    ARIUXDT ( RLDSAVE )
GENMOD  ARIUXDT
LOAD    ARIUXTM ( RLDSAVE )
GENMOD  ARIUXTM
```

Locally defined date/time formats on CICS OS/390 or VSE

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.

Accessing the DXT end user dialogs (ISPF only)

QMF's EXTRACT command accesses IBM's Data Extract (DXT) End User Dialogs. With these services, users can extract data from many different sources and load that data into DB2 tables. Possible data sources include IMS, VSAM, physical sequential data sets, and tables from other DB2 systems.

If you plan to support the EXTRACT command, ensure that:

- Version 2 Release 5 of DXT dialogs is operating at your installation.
- All potential users of the QMF EXTRACT command have been enrolled for DXT dialogs, and have been educated in its use.

For detailed information about DXT, see the appropriate DXT book listed in the bibliography at page Appendix H, "Bibliography", on page 775.

Supporting the EXTRACT command on OS/390

Another way to load data into tables is to use the DB2 Loader for sequential sources of data. See the*DB2 UDB for OS390 SQL Reference* manual for more information about the DB2 Loader.

To support the EXTRACT command, you must allocate data sets to each user of that command, and deallocate these data sets after the user ends the command.

The data sets can be in DXT libraries that are common to all users, or can be data sets created for the individual users enrolled in DXT.

The data sets are described in the *Data Extract: Planning and Administration Guide for Dialogs* manual. If you are enrolling DXT dialog users, you need that document. If you are not, all you need to know about the process is included in the following discussion.

Allocating resources on OS/390

QMF can support English, Kanji, and Uppercase (UCF) DXT dialogs. Each requires different DXT data sets, all of which can be allocated with ISPF LIBDEF statements (see "Allocating and deallocating resources using CLISTs" on page 403 for more information about using LIBDEF).

If you choose some other method of allocation, you can skip the following topics on modifying the CLISTs. The unmodified CLISTs will not interfere with the method you choose.

Allocating DXT data sets: Table 62 shows the data sets required for several DXT Version 2 Release 5 dialogs. The table identifies the data sets and their associated ddnames. For any given ddname, the data sets in the table are in addition to any data sets that were already allocated for that ddname.

The names shown in the table are the default names provided by DXT. Your installation might be using different names for these data sets.

In the table, each *n* is the language key. For DXT dialogs, the language keys are E (English), K (Kanji), and U (uppercase English).

Table 62. Data sets needed for DXT Version 2.5

DDNAME	Default Data Set Name
ISPLLIB	DXT250.DVRLOAD
ISPPLIB	DXT250.DVRPLIBn
ISPMLIB	DXT250.DVRMLIBn
ISPSLIB	*userid*.DXT250.DVRJEDIn DXT250.DVRSLIBn
ISPTLIB	*userid*.DXT250.DVRTLIBn DXT250.DVRTADMn
ISPTABL	*userid*.DXT250.DVRTLIBn
DVRDJEDI	*userid*.DXT250.DVRJEDIn or DXT250.DVRJEDIn (See note)
DVRDJED0	*userid*.DXT250.DVRJEDIn
DVRDIMEX	*userid*.DXT250.DVRIMEXn
DVREUADD	DXT250.DVRTADMn
DVRSTABL	DXT250.DVRTLIBn (See note)

Note: The library DXT250.DVRTLIBn applies only if your installation uses the DXT dialogs object-sharing capability.

Allocating and deallocating resources using CLISTs: **Note:** If you are not familiar with the ISPF LIBDEF statement, see the *ISPF V2 MVS Dialog Management Services* manual before reading further.

To allocate the needed data sets, you can either add JCL to your users' TSO logon procedures, or use two IBM-furnished CLISTs.

QMF calls one of these CLISTs just before issuing an EXTRACT command, and the other just after the command executes. With appropriate modifications, the first CLIST can allocate the added resources; the second can deallocate them. DSQABX1L is a sample CLIST that you can use to do the necessary allocation. This method is superior to adding JCL to your users' TSO logon procedures because it ensures that the DXT data sets are allocated only when they can be used.

Preparing the allocation CLIST: This CLIST is the member, DSQABX1L, of the library QMF720.SDSQCLTE. QMF calls this CLIST through the ISPF SELECT service whenever a user issues the EXTRACT command. The following modifications on DSQABX1L might be necessary before it can allocate:

1. Change the PROC statement.

   The original PROC statement is:
   ```
   PROC 0 DXTPRE(DXT250) LKEY(E) OBJSHR(NO)
   ```
   Because QMF does not pass parameters to the CLIST, you must ensure that the values for the following three keyword parameters are correct:

   DXTPRE
   Identifies the prefix for the DXT libraries.

   LKEY Identifies the language environment. It contains the language key. The original value, E, specifies the English-language environment.

   OBJSHR
   Can be YES or NO. The original value, NO, indicates that DXT object sharing is not in effect.
2. Remove the third executable statement.

   This is the statement `EXIT CODE(0)`, the last statement in the CLIST. It ensures that the CLIST does nothing if you aren't supporting the EXTRACT command or are making the allocations in some other manner.
3. Modify the code as necessary.

   Refer to the sample DSQABX1L CLIST in the 'QMF720.SDSQCLTE' sample library to see how the CLIST generates the data set names for its LIBDEF statements. These data set names are the defaults for DXT V2R5 dialogs. Modify the code, if necessary, to produce the names used at your installation, but do not modify the logic or the return codes for failed allocations.

Preparing the deallocation CLIST: The sample CLIST is the member DSQABX1F of the library QMF720.SDSQCLTE. QMF calls this CLIST through the ISPF SELECT service after the EXTRACT command runs. The following modifications of DSQABX1F might be necessary before it can deallocate:

1. Change the PROC statement.

   The original PROC statement is:

   ```
   PROC 0 QMFPRE(QMF720) LKEY(E) OBJSHR(NO)
   ```

   Because QMF does not pass parameters to the CLIST, be sure that the values for the following three keyword parameters are correct:

   QMFPRE
   Establishes a value in the CLIST for the variable `&QMFPRE`. This value is the first qualifier in a number of Version 3 Release 1 data set names. The original value, QMF720, is the installation default for QMF Version 7.2. If this qualifier is different at your installation, you might want to change the value for QMFPRE. Whether you need to do this depends on how you deallocate the data sets, as explained in the next steps.

   LKEY Identifies the language environment. Use the same value for LKEY that you did in the allocating CLIST.

   OBJSHR
   Indicates whether DXT object sharing is in effect. Use the same value for OBJSHR that you did in the allocating CLIST.

2. Remove the third executable statement.

   This is the statement `EXIT CODE(0)`. It ensures that the CLIST does nothing if you either aren't supporting the EXTRACT command or are making the allocations in some other manner.

3. If necessary, change the branching statement and modify the code.

   The third statement after the comments in the prolog is the branching statement `GOTO A`. Following this statement are two blocks of code: section A and section B. Both blocks deallocate the DXT data sets, but do so in different ways. If you choose section B to do the deallocation, you must change the branching statement to `GOTO B`.

   Section A nullifies every LIBDEF statement that the allocation CLIST issued. This deallocates the DXT data sets, but it might also deallocate data sets that the user needs after the EXTRACT command ends. Such data sets were allocated with LIBDEF statements before the user issued the EXTRACT command. If the user needs data sets that were allocated with LIBDEF statements issued before the EXTRACT command, then change the branching statement so that section B is invoked.

   For a data set to fit this description, a LIBDEF statement for its ddname must appear in this CLIST and in its allocating companion. For example,

the following LIBDEF statement adds a panel library, ABC.XYZ, to the libraries already allocated to the ddname ISPPLIB:

```
ISPEXEC LIBDEF ISPPLIB DATASET(ABC.XYZ)
```

In the allocation CLIST, this allocation disappears when the CLIST runs its LIBDEF statement for ISPPLIB. To restore it, you need to reissue the original LIBDEF statement in the deallocating CLIST. If this is the only allocation to restore, you might still use section A, just change the LIBDEF statement for ISPPLIB in that section.

The LIBDEF statements in section B reallocate QMF libraries to the ISPF ddnames. Change these statements to whatever is needed if you choose to use section B.

LIBDEF statements cannot deallocate data sets that were allocated through the TSO logon procedure or through TSO ALLOCATE statements. Therefore, you can always use section A if all the data sets needed for a QMF session are allocated in that way.

Supporting the EXTRACT command on VM

To support the EXTRACT command, you must make files available to the users of that command, and reallocate these files after a user ends the command.

These files do not appear in the QMF Invocation exec that is described in *Installing and Managing QMF on OS/390*. The file types can be in DXT libraries that are common to all users, or can be files created for the individual users when the users are enrolled in DXT.

The data files are described in the *Data Extract: Planning and Administration Guide for Dialogs* manual. If you are enrolling DXT dialog users, you need that document. If you are not, all you need to know about the process is included in the following discussion.

Allocating resources on VM

Table 63 on page 406 shows the files required for any variety of Version 2 Release 3 dialogs. The figure identifies the files and their associated FILEDEFs. For any given FILEDEF, the files in the table are in addition to any files that were allocated for that FILEDEF. The names shown in this table are the default names provided by DXT. Your installation might be using different names for these files. In the table, each lowercase letter n is the *language key*. For DXT dialogs, the language keys are E (English), K (Kanji) and U (Uppercase).

Example: For DXT dialogs in which the language key is E, the file name and file type to be added to ISPMLIB is named DVRMLIBE MACLIB.

Table 63. Files needed for Version 2 Release 5 DXT

FILEDEF	Default File Name/Filetype
ISPLLIB	DVRLOAD TXTLIB
ISPPLIB	DVRPLIBn MACLIB
ISPMLIB	DVRMLIBn MACLIB
ISPSLIB	DVRJEDIn MACLIB DVRSLIBn MACLIB
ISPTLIB	DVRTLIBn MACLIB DVRTADMn MACLIB
ISPTABL	DVRTLIBn MACLIB
DVRDJEDI	DVRJEDIn MACLIB
DVRDJEDO	DVRJEDIn MACLIB
DVRDIMEX	DVRIMEXn MACLIB
DVREUADD	DVRTADMn MACLIB
DVRSTABL	DVRTLIBn MACLIB

Allocating and reallocating resources using execs: There are two IBM-supplied execs that QMF calls. One is called just before the execution of an EXTRACT command, and the other just after execution ends. After modifying the execs, the first can allocate added resources, and the second can reallocate them.

DSQABX2L is a sample exec that you can use to do the necessary allocations. It has the following advantages over adding exec statements to your users' CMS invocation exec:

- It can apply to every user of the EXTRACT command.
- It does the allocations ONLY when a user issues an EXTRACT command.

Preparing the allocation exec: This exec is named DSQABX2L and is located on QMF's production disk. Whenever a user executes the EXTRACT command, QMF calls this exec through the ISPF SELECT service. The call passes the exec no parameters- a fact that is used when we consider possible exec modifications.

Before the exec can do its allocations, you must modify it. The following list describes some modifications that might or might not be necessary, and one modification that is mandatory:

1. Remove the first executable statement.

   This is the statement `EXIT 0`. It ensures that the exec does nothing if you aren't supporting the EXTRACT command or are making the allocations in some other manner.
2. Set the language key.

The first thing the exec does is to set the DXT language key variable (LKEY), to E for English. If your DXT product is not the English version, you must set the language key to the proper DXT value.

3. Set the object sharing variable.

   If you have taken advantage of the DXT dialogs object sharing capability, you need to set the variable OBJSHR to a value of YES. By doing this you allocate the shared variable DVRTLIB located on the DXT production disk. If you are not using object sharing, set the variable OBJSHR to a value of NO. Values for this variable can either be YES or NO.

4. Update the disk linkage.

   After setting LKEY and OBJSHR, the next thing that the exec does is to link to and access the DXT production disk. You might have to alter any or all of the following to fit your DXT installation:
   - DXT production disk owner ID
   - DXT production disk address
   - DXT production disk READ password
   - The QMF user's disk access address for the DXT disk
   - The QMF user's disk access mode for the DXT disk

5. Modify the code as necessary.

   Refer to the sample DSQABX2L exec on the QMF production disk to see how the exec generates the file names for its LIBDEF statements. These file names are the defaults. Modify the code, if necessary, to produce the names that are used at your installation, but do not modify the logic or return codes for failed allocations.

Preparing the reallocation exec: The DSQABX2F exec is located on the QMF production disk. QMF calls it through the ISPF SELECT service, right after the execution of the EXTRACT command. It is called to reallocate QMF libraries if the ISPF LIBDEF function was used to allocate DXT libraries. The call passes the exec no parameters, just as the call to the allocating exec passes that exec no parameters.

Before the exec can work properly for your users, you might need to modify it. If you allocated all your DXT libraries before you started QMF or ISPF, you should not modify this exec. It then exits without performing any library reallocation.

If you allocated QMF libraries using the ISPF LIBDEF function, you must execute this exec to reallocate the QMF libraries because they were replaced by DXT library definitions when the exec DSQABX2L was executed.

Possible modifications to the exec are:

- Remove the first executable statement.

This is the statement `EXIT 0`. It ensures that the exec does nothing if you are not supporting the EXTRACT command or are making the allocations in some other manner.

- If necessary, change the DXT disk address.

  The first thing the exec does is to release the DXT production disk. You need to modify the statement `USER_ADDRESS = '291'` depending on the changes you made when updating the disk linkage to DXT when executing the exec DSQABX2L.

Other allocation methods: Previously, we recommended that you use the exec for the DXT allocations and mentioned certain advantages for doing this. If you elect to use some other method of allocation, do not modify the exec. The unmodified exec will not interfere with your alternate method of allocation.

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 OS/390

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 OS/390, the resources are the QMF libraries. In the sample TSO logon procedure, these have names of the form:

  ```
  QMF720.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 the *Using QMF* manual.

  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 QMF720.SDSQCLTE on OS/390.
- The other components are members of QMF720.SDSQSAPE on OS/390.

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 the *Using QMF* manual.
- Place the renamed copy in QMF720.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 QMF720.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 QMF720.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 'QMF720.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 'QMF720.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" on page 410.

If you are using an NLF: Change the NLID in the member DSQAnD1S of the QMF720.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 QMF720.SDSQSAPE on OS/390. 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 QMF720.SDSQSAPE on OS/390.

  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 OS/390.

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 QMF720.DSQSAMP*n* library on OS/390.

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 *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 QMF720.SDSQCLTE. If you change the CLISTs or macros, change a copy, not the original, and place it in another library. On OS/390, 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 QMF720.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=QMF720.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 = QMF720.DSQSAMP&LANGCHAR
  SET &BASELIB = QMF720.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—QMF720.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 = 'QMF720.DSQPLIB&LANGCHAR'     /* ISPF Panel Library      */
 SET MNAME = 'QMF720.DSQMLIB&LANGCHAR'     /* ISPF Message Library    */
 SET SNAME = 'QMF720.DSQSLIB&LANGCHAR'     /* ISPF Skeleton Library   */
 SET LNAME = 'QMF720.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 QMF720. 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 22, "Customizing Your Start Procedure", on page 259.

Customizing the document editing interface on VM

The document interface is an IBM-supplied macro for the ISPF/PDF and XEDIT editors. 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.

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 execs and macros are on the QMF production disk on VM.
- The other components are on the QMF distribution disk on VM.

Renaming the document interface macro DSQAED2P

The ISPF/PDF macro component DSQAED2P is the macro that users call when they use the document interface. Give the macro a name that has more significance to your users. (Renaming this component has no affect on the other components.) Use the name GETQMF ISREDIT; this is the name used for the macro in this publication. In addition, the following should also be renamed:

DSQAED2X (an XEDIT macro), to GETQMF XEDIT
DSQAED2E (a REXX exec), to GETQMF exec

You should rename a copy rather than the original. You can place each renamed copy on the production disk where the original resides.

Placing the Q.DSQAED2S procedure in the database

The Q.DSQAED2S procedure is on the production disk. As the user Q, you can place it in the database by entering the following QMF commands:

```
IMPORT PROC FROM DSQAED2S PROC fm
SAVE PROC AS DSQAED2S (SHARE=YES
```

where `fm` is the QMF production disk.

If you are using an NLF: Save DSQA*n*D2S using the language key identifier for the language you want.

Transferring ownership to Q

If you cannot use QMF as the user Q, you can still issue these commands; however, the procedure is stored in the database under your own authorization ID, rather under Q. To give it the proper name, you must transfer its ownership to Q. You can do this by executing the following commands:

```
RUN  Q.DSQ0BSQI ( &T=Q.OBJECT_DIRECTORY, &N='DSQAED2S'
RUN  Q.DSQ0BSQI ( &T=Q.OBJECT_DATA, &N='DSQAED2S'
RUN  Q.DSQ0BSQI ( &T=Q.OBJECT_REMARKS, &N='DSQAED2S'
```

These commands execute an IBM-supplied parameterized query named Q.DSQ0BSQI. Each execution updates one of the QMF control tables. For these executions to be successful, you must have UPDATE authority on the three control tables, or some DB2 for VM authority that implies UPDATE authority.

If, for some reason, you cannot use the query Q.DSQ0BSQI, you can create a copy of it and use the copy instead. The copy would look like this:

```
 UPDATE Q.&T
  SET OWNER = 'Q'
  WHERE NAME = &N AND OWNER = USER
```

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 on the QMF distribution library on VM. Unlike the execs 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 on the QMF distribution disk on VM.

  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.

  Make sure that the new minidisk is accessed before the old one in the search order.

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: The DSQAnD0L component is on the NLF distribution disk and the messages are in the language set in the user's profile.

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 *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 execs and macros

As mentioned earlier, these components are all in the QMF production disk. If you change a component, change a copy, not the original, and place it in another library.

The minidisk must be accessed before the QMF production disk. If the document interface is issued from a current ISPF session, then that session needs to have the QMF and ISPF definitions for the ISPF libraries (the ones beginning with ISP) built. This is illustrated in DSQABD2I.

Changing DSQABD2Q: With the document interface, a user operating outside QMF can begin a QMF session. In that session, the user creates the report to be inserted into the current document. DSQABD2Q does the file definitions (FILEDEFs) for this session. Make whatever modifications to the exec you think necessary. For example, you might need to add FILEDEFs for files peculiar to your installation or you might have to change the links and accesses to the QMF, GDDM, and DB2 for VM disks.

Observe that some of these FILEDEFs involve GDDM files. The document interface does not itself use these files, but the user might find this necessary.

If you are using an NLF: Make a separate copy of DSQABD2Q to link to the QMF NLF production disk. Do not rename this exec.

Changing DSQABD2I: Ensure that the link and access to the ISPF/PDF disk is correct.

Changing DSQABD2C: This is the final component to be discussed. It can be modified as shown:

- Change the statement:

```
FILEDEF DSQPRINT PRINTER (LRECL 131 BLKSIZE 131 RECFM FBA)
```

- Change the statement:

```
ADDRESS ISPEXEC 'SELECT PGM(DSQQMF'LANG_CHAR')' ,
        'PARM (DSQSRUN='PROC_NAME') NEWAPPL(DSQ'LANG_CHAR')'
```

  This statement invokes QMF with the default DCSS name. (LANG_CHAR has the value E.) If the default DCSS is not being used, put the name in the PARM operand. For example, if you want to change the default DCSS name to QMFXXX, then the modified PARM operand would look like:

```
'PARM(QMFXXX(DSQSRUN='PROC_NAME'))...
```

- Change the statement:

```
ADDRESS COMMAND 'EXEC ISPSTART PGM(DSQQMF'LANG_CHAR')',
                'PARM(DSQSRUN='PROC_NAME') NEWAPPL'
```

  This statement invokes QMF with the default DCSS name. (LANG_CHAR has the value E.) If the default DCSS is not being used, put the name in the PARM operand. For example, if you want to change the default DCSS name to QMFXXX, then the modified PARM operand would look like:

```
'PARM(QMFXXX(DSQSRUN='PROC_NAME'))...
```

If you are using an NLF: Make a separate copy of DSQABD2C to specify the NLF DCSS name in the ISPSTART and SELECT QMF invocation statements. Do not rename this exec.

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 OS/390

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, QMF720.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 *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 110 on page 420 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 110. 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.

The EDIT command on VM

The following procedure assumes that you use an editor that can be called by an exec 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 an exec to invoke the editor, given the name of the file to be edited. For example, with the following command, QMF calls the exec, XYZEDIT, to edit the data set, `USERA.FILE`:

   ```
   XYZEDIT USERA FILE A1
   ```

2. Allocate the file USERA FILE A1 using the FILEDEF command specifying the file name of DSQEDIT. The FILEDEF needs to be allocated prior to invoking the editor. Therefore, the FILEDEF needs to be part of the QMF invocation process, or a FILEDEF needs to be established before invoking the EDIT command.

3. Instruct the users on how to invoke the editor through the EDIT command. A command would like like this:

   ```
   EDIT yyyy (EDITOR=xxxxxxxxx)
   ```

   where yyyy is either PROC or QUERY. Only the current procedure or query can be edited. xxxxxxxxx is the name of the exec created to invoke the editor.

   The file you can use can also be used for the ISPF/PDF editor.

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 in a file associated with the FILEDEF DSQEDIT. Remember that the edit transfer file described by the DSQEDI filedef cannot be associated to a disk that is used in the CMS shared file system. To bring the object into QMF, the user needs to issue a RESET DATA command.

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 26. 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 (for OS/390 and VSE) 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 either the appropriate *GDDM Installation and System Management* manual or the *GDDM System Customization and Administration* manual for GDDM 3.1.

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 OS/390

These services apply to native OS/390 batch, TSO, ISPF, and CICS

Native OS/390 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 425.

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 OS/390 batch, TSO and ISPF" on page 432.
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 OS/390 batch, TSO, and ISPF: In native OS/390 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 OS/390 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 the *GDDM System Customization and Administration* manual for GDDM or the appropriate *GDDM Installation and System Management* manual. These publications also provide more information on each type of GDDM device.

Creating the nickname specification

Here are the instructions to create nicknames on native OS/390, TSO, and CICS.

Native OS/390 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 111 for your ADMMNICK specification.

```
ADMMNICK NAME=nickname,TOFAM=family_type,DEVTOK=device_token(,TONAME=name)
```

Figure 111. 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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual for GDDM.
- 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 112. 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 112. Using the ADMMNICK specification to define a nickname for a Family 2 printer

Native OS/390 batch, TSO and ISPF: After you create your nickname in TSO, and native OS/390 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 (QMF720.DSQMAPE) that holds the map groups:

```
//ADMGGMAP DD DSN=QMF720.DSQMAPE,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 the *GDDM Installation and System Management* manual if you are using GDDM Version 2 Release 3 or the *GDDM System Customization and Administration* manual if you're using GDDM Version 3 Release 1.

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 OS/390 batch and TSO.

Native OS/390 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 113. Using the ADMMNICK specification to define a nickname for a Family 3 printer

After you create your nickname in TSO or native OS/390 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 114. 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 118 on page 433. 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 OS/390 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 115. 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 automatically through JES if you have the CSPOOL processing option set. For more information about Family 4 printing, see the*GDDM System Customization and Administration* manual.

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 the

GDDM System Customization and Administration manual for GDDM or the *GDDM Installation and System Management for OS/390* manual for GDDM.

- **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 the *GDDM System Customization and Administration* manual.

- **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 OS/390 batch, TSO, and CICS.

Native OS/390 batch, TSO and ISPF: In TSO, and native OS/390 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 the *GDDM System Customization and Administration* manual for GDDM or the *GDDM Installation and System Management for OS/390* for GDDM manual.

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 the *GDDM System Customization and Administration* manual for GDDM or the *GDDM Installation and System Management for OS/390* for GDDM manual.

Testing the nickname definitions in external default files for native OS/390 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 OS/390 batch, TSO and ISPF

For TSO, and native OS/390 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 116. Defining to CICS a nickname for a Family 2 GDDM printer

In VSE, all Family 1 and 2 devices must be described to CICS as queued.

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 117. Defining to CICS a nickname for a Family 3 GDDM printer

The*GDDM Installation and System Management for OS/390* manual for GDDM and the *GDDM System Customization and Administration* manual for GDDM describe 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 118.

```
* 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 118. 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 118 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.

GDDM services on VM

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 ADMDEFS PROFILE file or the GDDM defaults module, ADMADFV with the specifications of your nickname.
3. Update the PRINTER field of the user's row in the Q.PROFILES table.

Choosing a GDDM nickname for your printer

In CMS, GDDM searches the ADMDEFS PROFILE file and then the defaults module, ADMADFV, for a matching nickname that defines how and where to direct the output.

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 OS/390 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 the *GDDM System Customization and Administration* manual for GDDM or the appropriate *GDDM Installation and System Management* manual. These publications also provide more information on each type of GDDM device.

Creating a nickname specification

To create a nickname in CMS, you can add the nickname to your PROFILE ADMDEFS file. GDDM looks at this file first. If the nickname is not found, GDDM looks in the external default module, ADMADFV, in which you define a GDDM ADMMNICK specification.

Example nickname for a family 2 GDDM printer

To define the nickname GRAPHIC for a Family 2 GDDM printer, you might use an ADMMNICK specification similar to the one below. 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 119. Using the ADMMNICK specification to define a nickname for a Family 2 printer

After you create your nickname in CMS, a file with type ADMPRINT is created on your A-disk. This file has a file name of the printer that was supplied on input to the DSOPEN call. You can then print the ADMPRINT file using the ADMOPUV utility.

Example nickname for a family 3 GDDM printer

To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one below.

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S (CMS)
```

Figure 120. Using the ADMMNICK specification to define a nickname for a Family 3 printer

In CMS, a file with type ADMLIST is created. You can then send the formatted file to the printer you have chosen.

Example nickname for a family 4 GDDM printer

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 121. 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/VM if you have the CSPOOL processing option set. For more information about Family 4 printing, see the*GDDM System Customization and Administration* manual.

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 the *GDDM System Customization and Administration* manual for GDDM or the appropriate *GDDM Installation and System Management* manual for GDDM.

- **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 the *GDDM System Customization and Administration* manual.

- **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

In CMS, the ADMMNICK nickname specifications reside in the GDDM external defaults module ADMADFV, which is supplied with the GDDM product. The default module also contains default values for the GDDM product. The module is stored as a file with a type ASSEMBLE.

To update the modules with your nickname specification:

1. Copy the GDDM source file to your own storage.
2. Edit the source file to add the nickname.
3. Enter your ADMMNICK specification after the ADMMDFT statements in the module.
4. Reassemble and replace the changed default module. For more information on the default modules, see the *GDDM System Customization and Administration for GDDM* manual or the *GDDM Installation and System Management for VM for GDDM* manual.

Testing the nickname definitions in external default files

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.

Name the external default file ADMDEFS PROFILE and name the external default module ADMADFV. Testing the nickname definitions requires access to the minidisks containing these files. The external default file can be placed on any minidisk normally accessed when using QMF (for example the GDDM minidisks, which are accessed when using QMF).

GDDM uses external default modules more efficiently than a data set to find a given nickname.

GDDM services on VSE

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

In CICS, GDDM searches only the defaults module, ADMADFC. 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 OS/390 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 the *GDDM System Customization and Administration* manual for GDDM or the appropriate *GDDM Installation and System Management* manual. These publications also provide more information on each type of GDDM device.

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 below for your ADMMNICK specification.

```
ADMMNICK NAME=nickname,TOFAM=family_type,DEVTOK=device_token(,TONAME=name)
```

Figure 122. 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 the *GDDM System Customization and Administration* manual or the appropriate *GDDM Installation and System Management* manual.
- 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.

Example nickname for a family 3 GDDM printer

To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one below.

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S,TONAME=370P
```

Figure 123. 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. Make sure TONAME in the ADMMNICK specification and DESTID in the DCT have matching values.

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 the *GDDM System Customization and Administration* manual for GDDM or the appropriate*GDDM Installation and System Management* manual for GDDM.

- **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 the *GDDM System Customization and Administration* manual.

- **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

In CICS, the ADMMNICK nickname specifications reside in the GDDM external defaults module ADMADFC, which is supplied with the GDDM product.

To update the modules with your nickname specification:

1. Punch ADMADFC to ICCF or another editor, and edit the member to update it with the nickname specification.
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 the *GDDM System Customization and Administration* manual for GDDM or the *GDDM Installation and System Management for VSE* for GDDM manual.

In VSE, use the CEMT transaction to load a new copy of the ADMADFC phase into CICS storage. Use a statement similar to the following example:

```
CEMTS PROG(ADMADFC)NEW
```

Linking the nickname with a physical device

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 1 or 2 nickname with a physical device: For a family 1 or 2 printer, you can use macros to update CICS resource definitions in the TCT, or use CICS resource definition online (RDO) to update the CICS system definition (CSD) file.

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 in below.

```
GRAP     DFHTCT TYPE=TERMINAL,
                ACCMETH=VTAM,
                TRMIDNT=GRAP,
                TRMTYPE=SCSPRT,
                    . . .
                    . . .
                    . . .
```

Figure 124. Defining to CICS a nickname for a Family 2 GDDM printer

In VSE, all Family 1 and 2 devices must be described to CICS as queued.

Linking a family 3 nickname with a physical device: For a Family 3 printer in CICS/VSE, you need to update the DCT using macros. For example, for this nickname specification:

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=S3800N6,TONAME=S04E
```

you can update the CICS DCT.

Add the TYPE=SDSCI entry shown in Figure 125 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.

```
*********************************************************************
*           SYSTEM PRINTER FOR QMF OUTPUT.                          *
*           BLKSIZE:  132 + 1 FOR CTLCHR=ASA + 4 FOR RECFORM=VARUNB *
*********************************************************************
           DFHDCT TYPE=SDSCI,                                       +
                  BLKSIZE=137,                                      +
                  DSCNAME=UTMS04E,                                  +
                  RECFORM=VARUNB,                                   +
                  DEVADDR=SYS097,                                   +
                  DEVICE=1403,                                      +
                  TYPEFILE=OUTPUT,                                  +
                  CTLCHR=ASA
```

Figure 125. TYPE=SDSCI entry for the DCT

Add the TYPE=EXTRA entry shown in Figure 126 after all other TYPE=EXTRA and TYPE=INDIRECT DCT entries. The TYPE=EXTRA entry corresponds to the preceding TYPE=SDSCI entry by the matching value for DSCNAME.

```
********************************************************************** *
*           SYSTEM PRINTER FOR QMF OUTPUT.                           *
**********************************************************************
           DFHDCT TYPE=EXTRA,                                       +
                  DESTID=S04E,                                      +
                  DSCNAME=UTMS04E,RSL=1
```

Figure 126. TYPE=EXTRA entry for the DCT

Using QMF services to handle printing

Use this information for handling printing on native OS/390 batch and TSO, CICS, VM, and CICS/VSE.

Using QMF services for printing in native OS/390 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)
```

Term
Definition

Term
Definition

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.

Using QMF's DSQPRINT to handle printing on VM

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, or with a FILEDEF that points to the file or output class QMF uses for printing. The DD statement becomes part of your QMF startup EXEC, CLIST, or JCL. The FILEDEF is part of your QMF startup exec or is run from a QMF session using the QMF CMS command. You must allocate DSQPRINT before running the QMF PRINT command.

To add your printed output to PRINT FILE A, use the following syntax:

```
"FILEDEF DSQPRINT DISK PRINT FILE A(LRECL 133 BLKSIZE 133 RECFM V PERM",
         "DISP MOD"
```

The use of DISP MOD ensures that each PRINT command adds the latest print output to the end of the file, instead of overwriting the results of the previous PRINT command.

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(FBA) 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) (TSO)
"FILEDEF DSQPRINT PRINTER(LRECL 121 BLKSIZE 121 RECFM VBA PERM"(CMS)
```

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. For more information on DPRE, see "Displaying Printed Reports (DPRE) in TSO" on page xx and the *QMF Reference*.

Using QMF services to handle printing on VSE

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 VSE POWER LIST queue.

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 file, SYSLST, or SYSPCH. A transient data queue that you define as an intrapartition data queue is limited to 32,767 rows and can only be used to print data to a file.

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 to VSE POWER using QMF

You can print to a POWER-controlled printer using QMF. To do this, you need to write a program (see Figure xx on page 127) that segments the POWER output queue. You also need a QMF procedure to execute the QMF PRINT command and execute a CICS transaction that will execute the program. You can then run the QMF procedure using the PRINT PF key from the QMF Report screen.

You need to customize your CICS startup JCL, the DCT, and your synonym and function key tables to print using the LST POWER queue. (In your JCL and DCT entries, you may modify the references to POWER printer 04E, print class P, and logical unit SYS097, to values that are appropriate for your installation. Be sure that you make the modifications to all of the steps. Remember that printer 00E (usually print class A and logical unit SYSLST) is not recommended for QMF printing, because CICS statistics are routed there and would be segmented with your QMF report.

Modifying your CICS startup JCL: The following POWER LST statements need to be in the CICS startup JCL. 04E is the printer address for the QMF output:

```
*$$LST LST=04E,CLASS=P
```

Include the following assignment in the CICS startup JCL:

```
//ASSGNSYS097,04E
```

You do not need printer DLBLs/EXTENTs in the CICS startup JCL for the QMF printer destination.

Modifying your synonym and function key tables: Run the job shown in Figure 127 on page 448 to create and initialize two tables. The tables define a user-defined command synonym and a function key setting.

Ensure that column definitions that can contain nulls are not changed to NOT NULL, or QMF will not process the table entry properly.

```
* $$ JOB JNM=QMFPRINT,CLASS=0
* $$ LST CLASS=A
// JOB QMFPRINT SQL TABLES
// LIBDEF *,SEARCH=PRD2.SQL340
// EXEC ARIDBS
CONNECT Q IDENTIFIED BY ????????;    -- MODIFY PASSWORD
SET AUTOCOMMIT ON;
--
CREATE TABLE Q.UTM_SYN
  (VERB                CHAR(18)    NOT NULL,
   OBJECT              VARCHAR(31)         .
   SYNONYM_DEFINITION  VARCHAR(254)NOT NULL)
   IN PUBLIC.TEMP1;
--
CREATE UNIQUE INDEX Q.UTM_SYN_IDX1 ON Q.UTM_SYN (VERB, OBJECT);
--
GRANT SELECT ON Q.UTM_SYN TO PUBLIC;
--
INSERT INTO Q.UTM_SYN
  (VERB,SYNONYM_DEFINITION)
   VALUES ('SYSPRINT','RUN PROC Q.SYSPRINT_PROC');
--
CREATE TABLE Q.UTM_PFK
  (PANEL              CHAR(18)    NOT NULL,
   ENTRY_TYPE         CHAR(1)     NOT NULL,
   NUMBER             SMALLINT    NOT NULL,
   PF_SETTING         VARCHAR(254)        )
   IN PUBLIC.TEMP1;
--
CREATE UNIQUE INDEX Q.UTM_PFK_IDX1 ON Q.UTM_PFK
   (PANEL, ENTRY_TYPE, NUMBER);
--
GRANT SELECT ON Q.UTM_PFK TO PUBLIC;
--
INSERT INTO Q.UTM_PFK
   (PANEL, ENTRY_TYPE, NUMBER, PF_SETTING)
    VALUES ('REPORT', 'K',2,'SYSPRINT');
--
INSERT INTO Q.UTM_PFK
   (PANEL, ENTRY_TYPE, NUMBER, PF_SETTING)
    VALUES ('REPORT', 'L',1,
    '1=Help 2=Sys Print 3=End 4=Dflt Print 5=Chart 6=Query');
/*
/&
* $$ EOJ
```

Figure 127. User-defined command synonym and function key setting

Sample program to segment POWER output: The CICS program shown in Figure 43 on page 127 issues the POWER SEGMENT macro. You need to add a PCT entry for transaction S04E and a PPT entry for assembler program

SAMSEGMP. This is command-level assembler, so TWASIZE is zero in the PCT. This program segments the 04E POWER printer output.

```
* $$ JOB JNM=SAMSEGMP,CLASS=0,DISP=D
* $$ LST CLASS=A
// JOB SAMSEGMP ASSEMBLER CMD LEVEL CICS
// LIBDEF PAHASE,CATALOG=????.??????     -- your target library
* ************************************
* STEP 1: CICS COMMAND LEVEL TRANSLATION
* ************************************
// DLBL IJSYSPH,'ASM.TRANSLATION',0,SD
// EXTENT SYSPCH,1,0,??????,????       -- specify start, length
ASSGN  SYSPCH,DISK,VOL=DOSRES,SHR
// EXEC       DFHEAP1$,SIZE=512K
*ASM      XOPTS(NOEPILOG)
*********************************************************************
*                                                                   *
* PROGRAM NAME:  SAMSEGMP                                           *
* TRANSID:       S04E                                               *
*                                                                   *
* PURPOSE:       SEGMENT POWER-CONTROLLED PRINTER OUTPUT.           *
*                TRANSID IDENTIFIES THE DCT TYPE=EXTRA ENTRY        *
*                THAT CORRESPONDS TO A SPECIFIC SYSTEM PRINTER.     *
*                                                                   *
* NOTES:         STARTED BY QMFE, T0 RUN AS ASYNC TRANS.            *
*                THEREFORE, NO TERMINAL MESSAGES ARE DISPLAYED.     *
*********************************************************************
       EJECT
* ***************************************************************
* REGISTER USAGE
* ***************************************************************
*
R1     EQU    1
R2     EQU    2
R11    EQU    11              EXEC INTERFACE BLOCK
R12    EQU    12              PROGRAM BASE
R13    EQU    13              EIB DYNAMIC STORAGE
*
* ********************************************************************
* PROGRAM ENTRY
* ********************************************************************
*
SAMSEGMP DFHEIENT DTATREG=R13,CODEREG=R12,EIBREG=R11
*
         B    STRT                  ==> BRANCH AROUND
         DC   CLB'SAMSEGMP'         PROGRAM NAME.
         DC   CLB '01/20/94'        LAST MOD DATE.
STRT     DS   0H
         EXEC CICS ENQ RESOURCE(EIBTRNID) LENGTH(4).
         MVC  USERNM(8),BLANKS   * CLEAR SAVE AREA * 9/94
         EXEC CICS RETRIEVE INTO(USERNM) LENGTH(USERLEN)
         MVC  JOBJECL,JOBINIT
         MVC  JOBJECL+13(8),USERNM  * MOVE QMF SUERID TO JOB STATEMENT
*
         CLC  EIBTRNID,=C'S04E'     SEGMENT LST=04E ?
         BE   SEG04E
         DC   H'0'                  ABEND IF NOT 04E.
*
SEG04E   DS   0H
         SEGMENT DEVADDR=SYS097,JECL=JOBJECL
         SEGMENT DEVADDR=SYS097,JECL=S04EJECL
         B    DONE
DONE     DS   0H
```

Creating the QMF procedures: Create two procedures to send your report to a specified printer.

The first procedure is:

```
SET GLOBAL (Q='
RUN Q.SYSPRINT_PROC2
```

Save the procedure using SHARE=YES to enable your users to access the segmenting program.

```
SAVE PROC AS Q.SYSPRINT_PROC (SHARE=YES
```

Create the following two-line QMF procedure. The first line is a QMF command; the second line causes QMF to start the transaction S04E asynchronously. After QMF routes a report to the printer destination, transaction S04E causes POWER to segment the output and print it immediately. Without segmentation, you do not get the printout until CICS is shut down (like waiting for CICS STATS).

```
PRINT REPORT (QUEUENAME=S04E QUEUETYPE=TD LENGTH=62 WIDTH=132
CICS S043 (FROM=&Q&DSQAO_CONNECT_ID&Q)
```

Save the procedure using SHARE=YES:

```
SAVE PROC AS Q.SYSPRINT-PROC2 (SHARE=YES
```

Modifying your user's profile: Modify your user's profile to ensure that the SYNONYM column name is the name of the created synonym table (Q.UTM_SYN) and the PFKEY column name is the name of the created function key table (Q.UTM_PFK). To update one or more profiles in the QMF control table Q.PROFILE, run an SQL command. For example, to enable JONES to use the synonym, run the following command:

```
UPDAE Q.PROFILES
   SET SYNONYM = 'Q.UTM_SYN',
       PFKEYS  = 'Q.UTM_PFK'
       WHERE CREATOR = 'JONES'
```

Using your new print procedure: To produce the startup JCL and DCT changes, you need to cycle CICS. (Recycling CICS also replaces your old PCT and PPT tables with your changed tables, unless you used CEDA.) All other changes can be made before or after CICS is cycled, but we suggest that you do the table definition job and the segmentation program assembly before restarting CICS. Also, if a user is using QMF when the profile is updated, the changes will not take effect until the user exits QMF and starts a new session. You can test your changes as follows:

1. User logs on to QMF, runs a query, and the report screen is displayed
2. User sees customized function key line and presses PF2 (instead of PF4)
3. QMF associates PF2 with the synonym SYSPRINT

4. Synonym SYSPRINT becomes the command RUN PROC Q.SYSPRINT_PROC
5. Q.SYSPRINT_PROC issues a global, then invokes Q.SYSPRINT_PROC2
6. Q.SYSPRINT_PROC2 issues the print report command
7. Q.SYSPRINT_PROC2 also starts transaction S04E to segment printer
8. User sees message:

   ```
   OK, Your procedure was run.
   ```

Defining a synonym for the print function key

Use these instructions to define a synonym in native OS/390 batch, TSO, CICS, and VM.

Native OS/390 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.)

Defining a synonym for the print function key in VM

Here is a customization technique that allows a user to print an object without exiting QMF. The first two steps of this technique show how to define a command synonym for printing; the final step shows how to customize your Print function key. This technique can be used to invoke a local print utility when the Print function key is pressed.

1. Create a REXX exec that locally prints the current object. Here is a sample, called PRTQMF, using the QMF callable interface:

   ```
   /* PRTQMF REXX EXEC for local print utility called MPRINT */
   CALL DSQCI "PRINT PROC (PRINTER=MYPIRNT1"
   mprint MYPRINT1 ADMLIST A
   ```

   This example assumes you have a MYPRINT1 nickname defined and that it creates a file with a file type of ADMLIST.

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 DSPRINT *?
CALL DSQCIX "EXPORT PROC TO MYPROC"
mprint MYPROC PROC A
```

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', 'CMS 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 a key for each panel. a query to customize function key 4 on the procedure panel would look 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 on VSE

For an example of using a command synonym, see "Modifying your synonym and function key tables" on page 447.

Printing objects

The rules for printing QMF and database objects vary, depending on the type of object. Table 64 summarizes the requirements for each object.

Table 64. Summary of print requirements for QMF and database objects

Object type	Nickname required	GDDM gets control	Where output is routed for OS/390 or VM	Where output is routed for VSE
Chart	Yes	GDDM ICU always gets control when the PRINT command is issued.	Output is controlled by GDDM.	Destination associated with TONAME in the ADMMNICK specification.
Form	Yes	GDDM always gets control when the PRINT command is issued.	Output is controlled by GDDM.	Destination associated with TONAME in the ADMMNICK specification.
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.	Temporary storage or transient data queue. If a Nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.

Table 64. Summary of print requirements for QMF and database objects (continued)

Object type	Nickname required	GDDM gets control	Where output is routed for OS/390 or VM	Where output is routed for VSE
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.	Temporary storage or transient data queue. If a Nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
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.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
Prompted query	Yes	GDDM always gets control when the PRINT command is issued.	Output is controlled by GDDM.	Destination associated with TONAME in the ADMMNICK specification.
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.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
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.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
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.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.

Chapter 27. 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 OS/390, TSO, CMS, or CICS on VSE 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.

Default synonyms on OS/390

Workstation database server users: After installation, these synonyms appear in the table Q.COMMAND_SYN_TSO.

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 the *QMF Reference* manual.

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 the *Using QMF* and *Developing QMF Applications* manuals.

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 the *Using QMF* manual.) 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 QMF720.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 QMF720.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)"
  ```

Default synonyms on VM

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 the *QMF Reference* manual.

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 the *Using QMF* and *Developing QMF Applications* manuals.

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. You must start QMF under ISPF in order to use the above three applications.

Displaying printed reports (DPRE) in CMS

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 the *Using QMF* manual.) 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 CMS are a procedure named Q.DSQAER2p in the database and a CLIST named DSQABR23 in the QMF production library.

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 CMS FILEDEF statement for DSQPRINT. If 132 is too small, the CMS FILEDEF 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 name of the file 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 file that holds the material to be browsed. If an error stops the run, this definition might still be in effect after the run terminates.

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 CMS, you can change the parameters that DSQPRINT has when DPRE ends normally. This is controlled by statements in the QMF procedure DSQAER2P which is invoked by the DPRE command synonym. This statement:

```
Print_opts = "LRECL 121 RECFM FBA BLKSIZE 1210"
Address COMMAND "FILEDEF DSQPRINT PRINTER ("Print_opts
```

changes the record format (LRECL) from 133 to 121, and changes the block size (BLKSIZE) from 1330 to 1210.

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 OS/390

Use the following procedure to create a command synonym table. Then see "Entering command synonym definitions into the table" on page 465 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 OS/390 the storage container for a table is referred to as a table space. The examples below use the OS/390 default table space name, TBSPACE1. Refer to the appropriate *DB2 System Administration* manual for how to add a table space. 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 129. 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 130 on page 463 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 130. 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 130.

   ```
   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 OS390 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.

Creating a command synonym table on VM and VSE

Use the following procedure to create a command synonym table. Then see "Entering command synonym definitions into the table" on page 465 for instructions on entering your synonyms and their definitions.

1. If necessary, acquire or add a dbspace to hold the command synonym table.
2. From the QMF SQL query panel, run an SQL CREATE TABLE statement similar to the one in Figure 131 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 DBSPACE1
```

Figure 131. 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 SYSTEMS.SYSCATALOG table using the following example for the COMMAND_SYNONYMS table created with the query in Figure 131.

   ```
   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 SYSTEMS.SYSCATALOG table.

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 132 to enter your synonyms into the table. You can also use the Table Editor to update the table, as explained in *Using QMF*.

```
INSERT INTO COMMAND_SYNONYMS (VERB,OBJECT,SYNONYM_DEFINITION)
  VALUES('COMPUTE', 'MONTHLY_SALES', 'RUN PROC JONES.SALES_FIGURES')
```

Figure 132. Creating a command synonym definition

After it is activated according to the procedure in "Activating the synonyms" on page 474, the synonym COMPUTE MONTHLY_SALES runs a QMF linear procedure called SALES_FIGURES, owned by user JONES.

The query in Figure 133 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 133. 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 132 and 133 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, TSO, or CMS 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 didn't 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 the *QMF Reference* manual to find out if the command requires a parameter.

OS/390 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.

VM concerns

The verb is your own word for the QMF RUN command or CMS 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.

VSE concerns.

The verb is your own word for the QMF RUN command or CICS 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 18 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)

“User x”.“Net Sales”
JANUARY__PRODUCTIVITY (over 18 characters)

“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 OS/390

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 the *Developing QMF Applications* manual.

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 134 on page 469 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 134 on page 469, 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 134 on page 469, 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 134. 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 135:

```
SYNONYM
VERB           OBJECT           DEFINITION
-----------    --------------   ------------------
SHOW           SALES            RUN PROC SALES_PROC
```

Figure 135. 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.

Choosing the synonym definition on VM

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 CMS command that calls a QMF procedure.

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 the *Developing QMF Applications* manual.

Using a procedure in the synonym definition: Your synonym definition can include a linear procedure that does the work of several QMF commands. For example, the procedure in Figure 137 on page 470 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 a file.
3. Runs a QMF procedure to route the report to a predefined print destination. Your definition for a synonym that runs this procedure might look similar to this:

```
-- Procedure name: SALES_PROC
 RUN QUERY SALES_DATA
 CMS FILEDEF DSQPRINT DISK MYPRT FILE A (LRECL 75 BLKSIZE 75 RECFM F
 PRINT REPORT (PRINTER=' '
```

Figure 136. Sample procedure to run using a command synonym ON CMS

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. The DSQEC_NLFCMD__LANG variable is discussed in "Enabling English support in an NLF environment" on page 421

```
SYNONYM
VERB          OBJECT          DEFINITION
-----------   --------------  ------------------
SHOW          SALES           RUN PROC SALES_PROC
```

Figure 137. Using a command synonym to run a procedure

Choosing the synonym definition on VSE

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 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 the *Developing QMF Applications* manual.

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 138 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 CICS temporary storage. In Figure 138, XYZ is the name of the temporary storage queue.
3. Runs a CICS transaction to route the report from temporary storage to a predefined print destination. In Figure 138, 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)
 CICS RPTX (FROM=('REPORTX, XYZ'))
```

Figure 138. 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 139:

```
SYNONYM
VERB           OBJECT           DEFINITION
-----------    ---------------  ------------------
SHOW           SALES            RUN PROC SALES_PROC
```

Figure 139. 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. The DSQEC_NLFCMD__LANG variable is discussed in "Enabling English support in an NLF environment" on page 421

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 140 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 140. 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 141 shows an example of a synonym defined using &ALL.

```
SYNONYM
VERB        OBJECT     DEFINITION
----------  ---------  ------------------------
SHOW_INFO   -          RUN QUERY STAFFQUERY (&ALL)
```

Figure 141. 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 the *QMF Reference* manual.

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 141 on page 472, 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 the *QMF Reference* manual.

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 can't distinguish the command synonym EXECUTE from the base QMF command EXPORT. See the *QMF Reference* manual for the proper abbreviations for QMF commands.

Activating the synonyms on OS/390

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 142:

```
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 142. 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 142 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" on page 476.

3. Instruct users to end the current QMF session and start another to activate the new synonyms.

Activating the synonyms on VM and VSE

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 below:

```
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='CMS'
        AND ENVIRONMENT='CMS'
```

Figure 143. 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 143 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 use the QMF CONNECT command to reconnect to the database to activate the new synonyms. For example, user JONES who has the password MYPW needs to enter:

```
CONNECT JONES (PA=MYPW
```

Each time you make a change to the table, instruct users to reconnect to the database to activate the changes you made.

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 appropriate *DB2 UDB 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 144 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 144. 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 145 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 145. 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 146 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 146. 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 147 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 147. Creating a view that uses an extra table to control use of synonyms

Chapter 28. 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

Key	Executed Command
Comments	SWITCH COMMENTS
Condition	SPECIFY CONDITION
Delete	DELETE
Describe	DESCRIBE
End	END
Exit	END
Forward	FORWARD
Help	HELP
Index	HELP INDEX
Keys	HELP KEYS
List	LIST
Menu	HELP MENU
More Help	HELP MORE
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 the *QMF Reference* manual.

Creating the function key table

Use these instructions to create the function key tables on OS/390, VM, and VSE.

Creating the table on OS/390

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 148 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 148. Creating a function key table

   See the appropriate *DB2 UDB 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 OS390 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 the *DB2 UDB for OS390 SQL Reference* manual.

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.

Creating the table on VM and VSE

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 149 to create the table. Substitute your own name for MY__PFKEYS. Substitute your own dbspace name for SPACE1.

```
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 DBSPACE1
```

Figure 149. Creating a function key table on VM and VSE

2. Add comments to the SYSTEM.SYSCATALOG table 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 SYSTEM.SYSCATALOG table.

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 65 on page 485.

Table 65. 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 488 shows the IDs you need to use for full-screen panels. "Window panel identifiers" on page 489 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 don't 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're changing the definition for F5, enter a 5 in this column.
F__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 doesn't 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 66. 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 484.
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 484). 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 150 on page 487 for an example. For window panels, you need to type only the label of the key in this column. See Figure 151 on page 488 and Figure 152 on page 488 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 488. 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 150 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 150. 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 151 on page 488 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 151. 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 152 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 152. 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 153 on page 489. 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 153. 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'll 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

Use these instructions to activate new function key definitions on OS/390, VM, and VSE.

Activating definitions on OS/390

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 154 on page 492 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 154. 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 476. 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.

Activating definitions on VM

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 155 on page 493 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 = 'CMS'
        AND ENVIRONMENT = 'CMS' (or 'TSO')
```

Figure 155. Making customized function keys accessible to a user

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 476 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 reconnect to the database to initialize a QMF session with the new function key definitions. For example, user JONES who has the password MYPW needs to enter:

   ```
   CONNECT JONES (PA=MYPW
   ```

Activating definitions on VSE

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 156 on page 494 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 = 'CICSVSE'
        AND ENVIRONMENT = 'CICSVSE' (or 'TSO')
```

Figure 156. Making customized function keys accessible to a user

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 476. 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 reconnect to the database to initialize a QMF session with the new function key definitions. For example, user JONES who has the password MYPW needs to enter:

   ```
   CONNECT JONES (PA=MYPW
   ```

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 29. 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.

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. The *QMF Reference* manual 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 67.

Table 67. 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 the *Using QMF* manual. 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* manual.

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 68.

Table 68. 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.
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 69 on page 505.

Calling your exit routine to format the data

Use these instructions to call your exit routine on OS/390, VM, and VSE.

Calling your exit routine on OS/390

Figure 157 shows how QMF and your edit exit routine work together to format data using the edit codes you define.

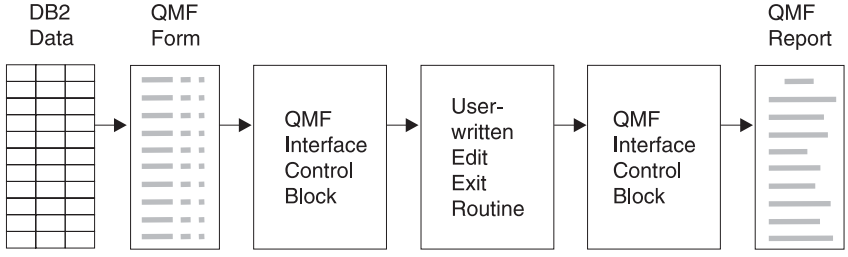


Figure 157. How a user edit routine works with QMF for OS/390

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 505. 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 QMF720.SDSQSAPE.

Language	TSO and native OS/390 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 QMF720.SDSQSAPE on OS/390.

QMF supplies the user edit routine DSQUEDIT for TSO and native OS/390, and a reentrant module, DSQUECIC, for CICS, which are located in the QMF

library QMF720.SDSQLOAD. Delete or rename the QMF-supplied module when you are ready to use your edit routine.

See Figure 159 on page 502for the general structure of an edit routine.

Calling your exit routine on VM

Figure 158 shows how QMF and your edit exit routine work together to format data using the edit codes you define.

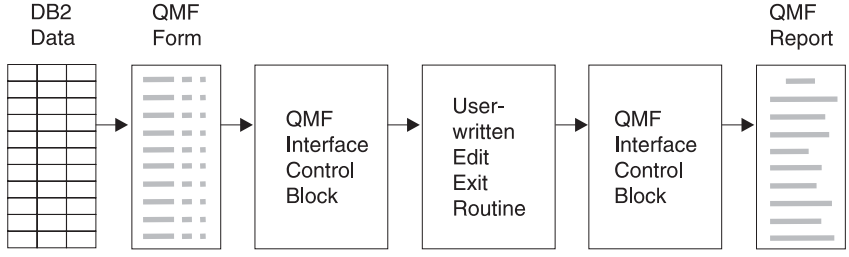


Figure 158. How a user edit routine works with QMF for VM

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 505. QMF also passes into the input area the data to be formatted and an output area that holds the formatted result.

IBM supplies three different versions of a sample edit exit routine. One version is for assembler (DSQUXDTA), one is for PL/I (DSQUXDTP), and one is for COBOL (DSQUXDTC).

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 on the QMF production disk on CMS.

QMF supplies the user edit routine, DSQUEDIT, for CMS. DSQUEDIT is a relocatable module file and a text file on the QMF production disk. Delete or rename the QMF-supplied module and text file when you are ready to use your edit routine.

VM Note: The use of a relocatable module file facilitates user edit code development because a module file on the user's A-disk can be tested without renaming or deleting the QMF-supplied user edit routine from the QMF production disk. This reduces the impact on other QMF users. Once you have written and assembled or compiled your edit routine, you need to consider the method of making your routine available to QMF for execution. The user edit routine can be executed in text or module format. The use of a relocatable CMS module file is the preferred method of generating a user edit routine.

When QMF for VM is started, QMF attempts to load the edit routine as follows:

1. Issue CMS NUCXLOAD for DSQUEDIT. NUCXLOAD loads a CMS module file that has relocation information saved, or as a member of an OS load library.
2. Issue OS LOAD (SVC 8) for DSQUEDIT. If use of NUCXLOAD is not successful, QMF then issues an OS LOAD (SVC 8). OS LOAD loads a text file, a member of a TXTLIB, or a member of an OS load library.

Different versions of the interface control block are used for Assembly, PL/I, and COBOL edit routines. However, the fields of the control block and the input they contain are the same regardless of the programming language the routine is written in. Figure 159 shows the general structure of the edit routines.

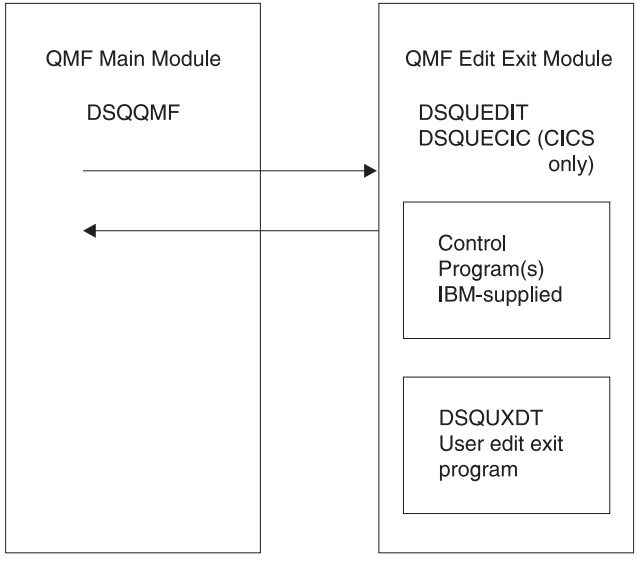


Figure 159. General program structure for edit routines

Calling your exit routine on VSE

Figure 160 shows how QMF for VSE and your edit exit routine work together to format data using the edit codes you define.

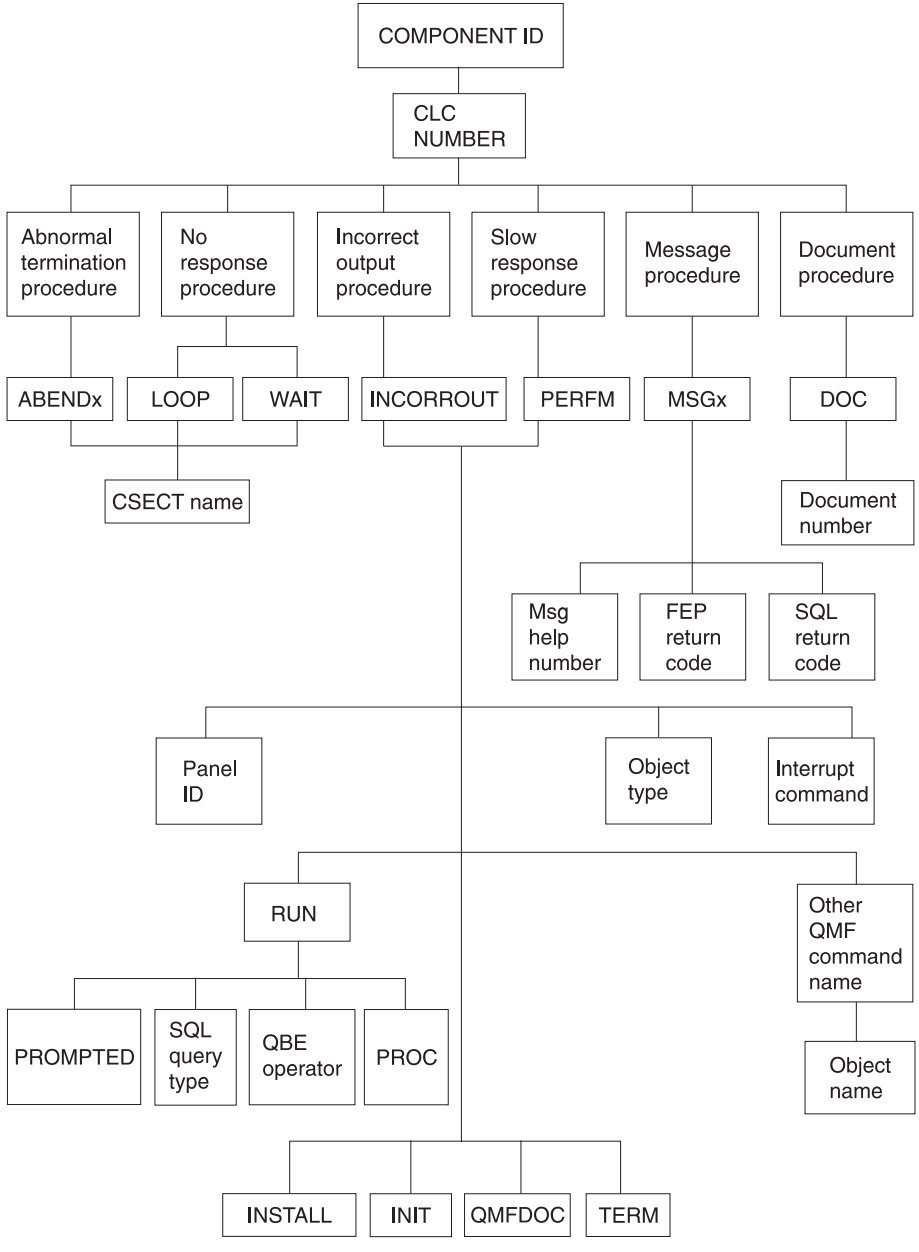


Figure 160. How a user edit routine works with QMF for VSE

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 505. QMF also passes into the input area the data to be formatted and an output area that holds the formatted result.

IBM supplies three different versions of a sample edit exit routine. One version is for assembler (DSQUXDTA), one is for PL/I (DSQUXDTP), and one is for COBOL (DSQUXDTC).

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 the QMF sublibrary on VSE.

The DSQUECIC program supplied with QMF is a sample meant to be used with the sample edit programs. Because of this, the program simply returns an error code when it is called, and QMF displays a message indicating you attempted to use an unsupported edit code.

After you write your edit exit program, name is DSQUECIC. Then translate, assemble (or compile), and link-edit the program to form the edit exit phase named DSQUECIC. You need to replace the IBM-supplied program with your new program. Do not change the name of the program; it remains DSQUECIC.

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 69.

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 Assembly, 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 69. 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 69. 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.

Table 69. Fields of the QMF interface control block (continued)

Name	Contents
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.
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.

Table 69. Fields of the QMF interface control block (continued)

Name	Contents
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 498.

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 OS/390, TSO, CICS, and CMS.

Writing an edit routine for native OS/390, 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 161 on page 510 shows the program structure of an Assembler edit exit routine for native OS/390, TSO, or ISPF.

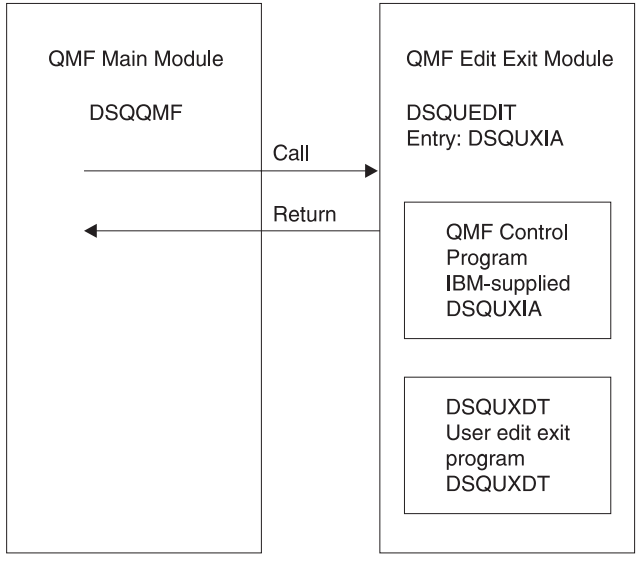


Figure 161. Program structure of an Assembler edit exit routine for TSO, native OS/390,

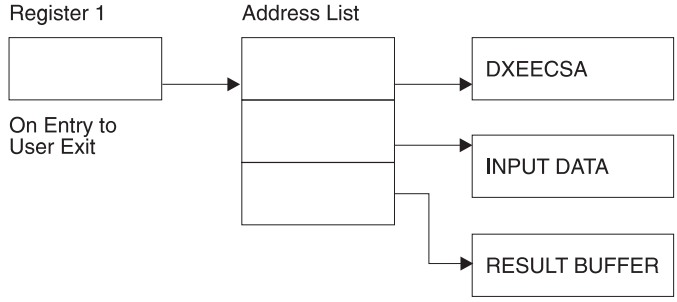
Example program DSQUXDTA

The IBM-supplied sample edit program for Assembler, DSQUXDTA, is located in the QMF720.SDSQSAPE library for OS/390. 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 OS/390. DXEECSA defines the input fields, giving them the names we are using in this chapter.

How an Assembly edit routine interacts with native OS/390

The user edit program is called as a subroutine in TSO and native OS/390 using a standard Assembly 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 Assembly DSECT for DXEECS is shipped with QMF as DXEECSA, located in library QMF720.SDSQUSRE on OS/390 or in DSQUSERE MACLIB on CMS. Include this DSECT in your program using the Assembly 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 Assembly 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 OS/390

During the assembly, QMF edit exit interface control block DXEECSA, located in QMF sample library QMF720.SDSQUSRE in TSO or native OS/390, 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 QMF720.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 OS/390

The following are example statements for assembling and link-editing your job for TSO or native OS/390:

```
//sampasm    JOB
//STEP1      EXEC PROC=ASMHCL
//* Provide Access to QMF Edit Macro DXEECSA
//C.SYSLIB   DD  DSN=QMF720.SDSQUSRE,DISP=SHR
//C.SYSIN    DD  *
          .
          Your program or copy of QMF sample DSQUXDTA
          .
/*
//* Provide Access to QMF Interface Module
//L.QMFLOAD  DD DSN=QMF720.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 162 shows the program structure of an Assembler edit exit routine for CICS.

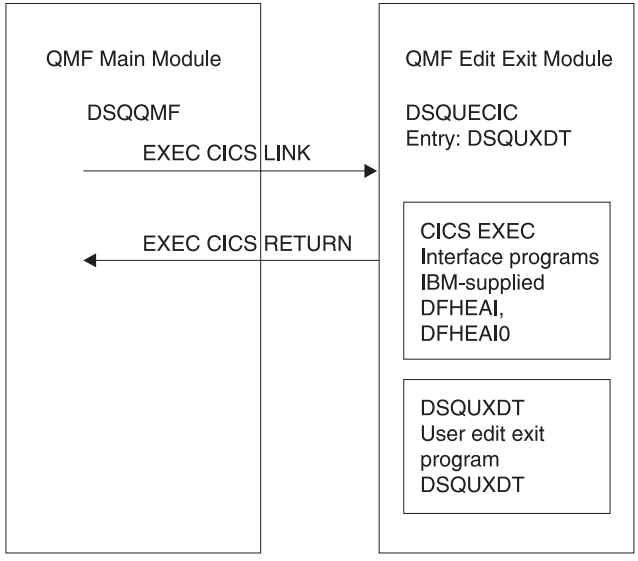
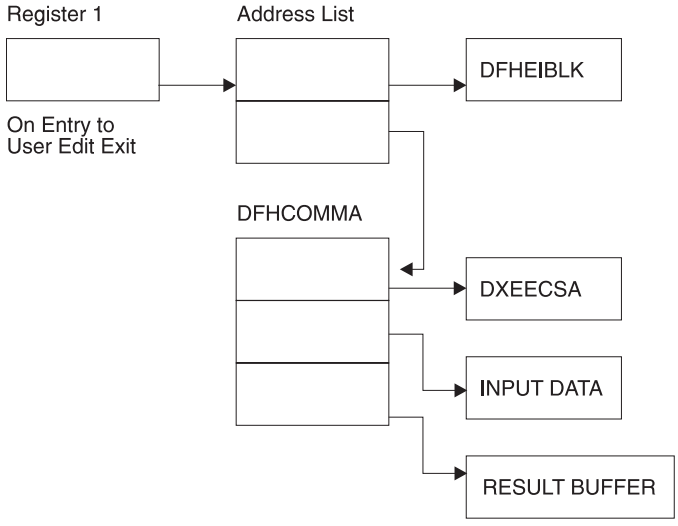


Figure 162. Program structure of an Assembler edit exit routine for CICS

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 QMF720.SDSQSAPE. Include this DSECT into your program using the Assembly 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 QMF720.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 OS/390

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='QMF720.SDSQLOAD'
//TRN.SYSIN DD *
      .
      Your program or modified copy of QMF sample DSQUXCTA
      .
/*
//* Provide access to QMF Edit Macro DXEECSA
//ASM.SYSLIB DD  DSN=QMF720.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 QMF720.SDSQSAPE on OS/390. 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 Assembly version of the DXEECS control block. This file is located in library

QMF720.SDSQSAPE on OS/390, on the QMF production disk on CMS, or in the QMF sublibrary as DXEECSA.A on VSE.

Writing an edit routine for VM

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 163 shows the program structure of an Assembly edit exit routine for VM.

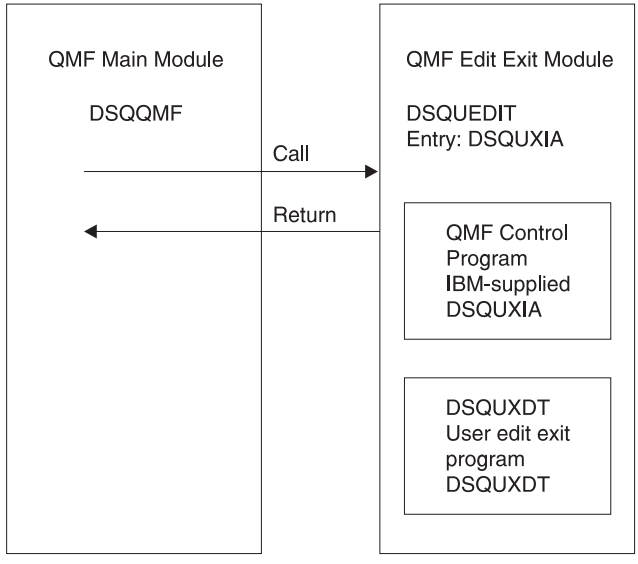


Figure 163. Program structure of an Assembler edit exit routine for VM

Example program DSQUXDTA

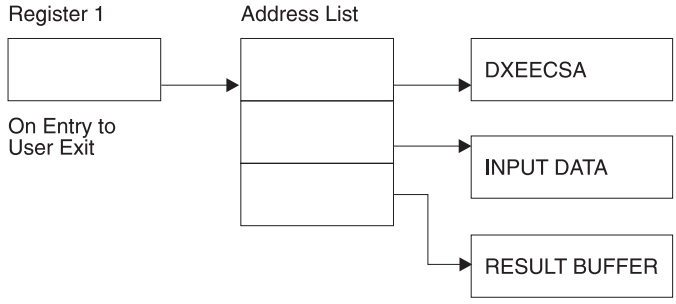
The IBM-supplied sample edit program for Assembler, DSQUXDTA, is located in the QMF production disk. The sample program is commented so that you can modify it to suit your needs. If you plan to use this example program, copy it to your program library and change its name to DSQUXDT. Near the bottom of this file is a COPY statement for DXEECSA. which is a member of DSQUSERE MACLIB on OS/390. DXEECSA defines the input fields, giving them the names we are using in this chapter.

How an Assembler edit routine interacts with CMS

Linkage obeys the standard IBM calling conventions. On entry to your edit exit program, the following conditions exist:

- The parameter list contains three four-byte addresses. The addresses point to:

- the control block
- the value to be formatted
- the storage reserved for the formatted results



- 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 DSQUSERE MACLIB on CMS. Include this DSECT in your program using the Assembly COPY statement.

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 generating your program on CMS

Before you assemble your program, ensure that you can access the IBM-supplied control block DXEECSA, which is located in the QMF library DSQUSERE MACLIB on the QMF production disk. You need to access the QMF production disk and issue the CMS command GLOBAL MACLIB for the QMF macro library. For example:

```
GLOBAL MACLIB DSQUSERE
```

Assemble your edit program, DSQUXDT, using HLASM or the Assembly supplied with CMS.

Before you create the DSQUEDIT module file to generate your program, ensure that you can access the IBM-supplied control module (DSQUXIA). DSQUXIA is located on the QMF production disk. You need to access this disk prior to creating the module file.

To create the DSQUEDIT module file, use the CMS LOAD and GENMOD commands as follows:

1. Load the text files that make up the DSQUEDIT module. The DSQUEDIT module must be relocatable. To be relocatable, the module must be loaded with RLD entries. You do this by specifying the RLDSAVE option on the CMS LOAD command. The entry point to the DSQUEDIT module must be DSQUXIA. Issue the following CMS LOAD command:

   ```
   LOAD DSQUXIA DSQUXDT (RLDSAVE RESET DSQUXIA)
   ```

   You can run your edit routine in either 24-bit or 31-bit addressing mode. QMF manages address switching as required. You can specify 31-bit addressing on the CMS LOAD command. For example:

   ```
   LOAD DSQUXIA DSQUXDT (RLDSAVE RESET DSQUXIA AMODE 31 RMODE ANY)
   ```

2. Generate the DSQUEDIT module. Issue the CMS GENMOD command to generate the DSQUEDIT module from the text files just loaded by the CMS LOAD command:

   ```
   GENMOD DSQUEDIT (AMODE 31 RMODE ANY)
   ```

Once the user edit routine is tested it can be placed on the QMF production disk or user disk that is available when you start QMF.

Writing an edit routine in Assembler for CICS/VSE

The QMF edit exit interface for Assembler in CICS consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSA.A Use a COPY statement to include the interface control block in your source deck.

- CICS prolog macro DFHEIENT, which is shipped with CICS. The CICS epilog macro DFHEIRET is not needed for an EXEC CICS RETURN.
- CICS command interface modules, which are shipped with CICS as DFHEAI and DFHEAI0.
- Your edit exit program, which is named DSQUECIC

Figure 164 shows the program structure of an Assembler edit exit routine for CICS.

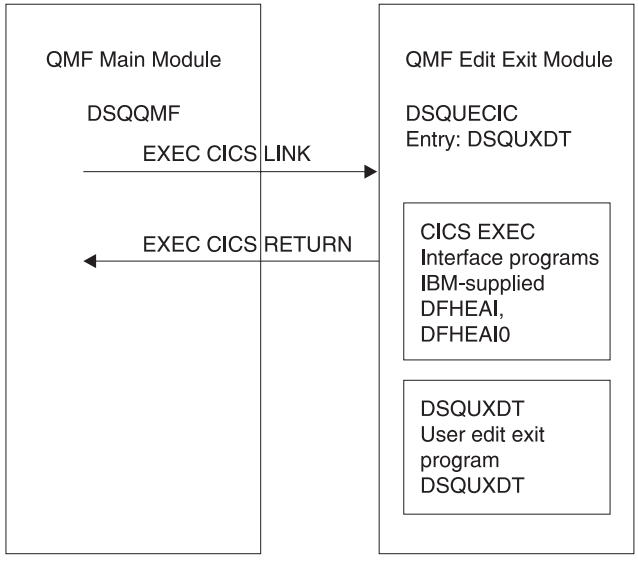


Figure 164. Program structure of an Assembler edit exit routine for CICS

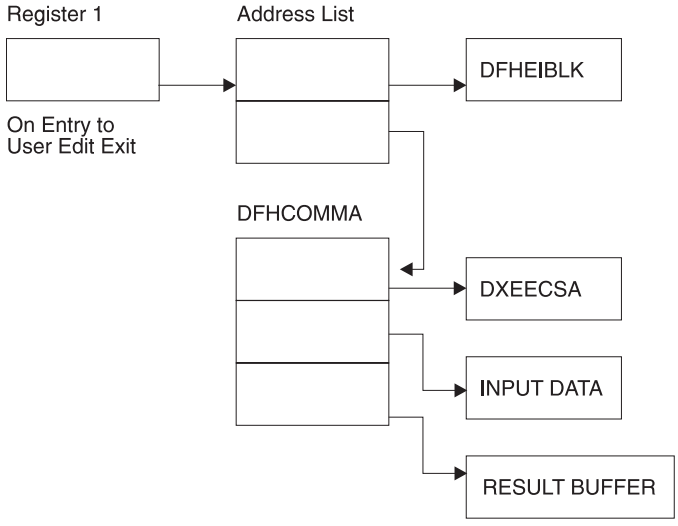
Example program DSQUXCTA

The IBM-supplied example edit program in Assembler, named DSQUXCTA.A, is located in the QMF sublibrary on VSE as DSQUXCTA.Z. 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 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.



- Register 13 contains the address of a standard CICS working storage area as described by CICS supplied macro DFHEISTG.

An Assembly DSECT for DXEECS is shipped with QMF as DXEECSA.A, located in the sublibrary where QMF is installed. Include DXEECSA.A into your program using the Assembler COPY statement.

Return control to QMF by using the standard CICS RETURN command.

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.

Refer to the DXEECSA file provided by QMF as a sample Assembler version of the DXEECS control block. This file is located in the QMF sublibrary as DXEECSA.A on VSE.

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.

Return control to QMF by using the CICS RETURN command; for example, EXEC CICS RETURN.

Assembling your program on VSE

When you assemble your program, ensure the LIBDEF search chain includes the CICS and QMF sublibraries so that the CICS macros and the edit exit interface control block (DXEECSA.A) can be found. Use the following Assembler compiler options to assemble the routine:

```
'LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXIT(ORDER=EA)))'
```

These compiler options require that you specify an E-deck exit. EDECKXIT is a library exit for Assembler that enables the processing of E-decks. This exit is required here to process CICS E-decks.

VSE/ESA provides a skeleton to help you set up the E-deck exit. You can use the skeleton without modifying it; however, before you use the skeleton, ensure you enable the exit according to instructions provided in the *VSE Guide to System Functions*.

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 VSE

Figure 67 on page 174 shows the sample job DSQ3XCTA.Z, which is shipped with QMF. This job translates, compiles, and link-edits the example Assembler program (DSQUXCTA.Z), which is also shipped with QMF. Use the sample job as a starting point to create JCL that translates, assembles, and link-edits your own edit exit routine. For more information on installing an Assembler program in CICS, see the *CICS System Definition Guide*.

```
// JOB DSQ3XCTA Install  QMF Edit Exit for COBOL
*  ------------------------------------------------------------------------
*  Install QMF Edit Exit (HLASM)
*  ------------------------------------------------------------------------
// SETPARM VOLID=volid   *-- update volid for syspch
//  SETPARM START=rtrk     *-- update start track/block (syspch)
//  SETPARM SIZE=ntrks   *-- update number of tracks/blocks (syspch)
*-------------------------------------------------------------------------
//  DLBL   IJSYSPH,'ASM.TRANSLATION',0
//  EXTENT SYSPCH,,1,0,&STARTL,&SIZE.
    ASSIGN SYSPCH,DISK,VOL=&VOLID.,SHR
*   Library search chain must contain the QMF, CICS and HLASM sublibraries
//  LIBDEF *,SEARCH=(PRD2,PROD,PRD1.BASE,PRD2.CONFIG)
//  LIBDEF PHASE,CATALOG=PRD2.PROD
*-------------------------------------------------------------------------
//  STEP 1: Translate Edit ExitQMF720 program
*-------------------------------------------------------------------------
//  EXEC    DFHEAP1$
    :
    Assembly source program here
    :
/*
*-------------------------------------------------------------------------
*   Step 2: Assemble Edit Exit program
CLOSE SYSPCH,00d
// DLBL   IJSYSIN,'ASM,TRANSLATION',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=&VOLID.,SHR
// OPTION CATAL,DECK,SYM,ERRS

   PHASE DSQUECIC,*,SVA

         INCLUDE DFHEAI
         INCLUDE DFHEAI0

// EXEC ASMA90,SIZE=(ASMA90,50K),                                    C
               PARM='LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXITC
               (ORDER=EA)))'
CLOSE SYSIPT,SYSRDR
/*
*-------------------------------------------------------------------------
*  Step 3: Link-dit Edit Exit program
*-------------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SY;SPCH,00D      IF 1A93D, CLOSE SYSPCH,00D
/&
```

Figure 165. Example JCL for translating, assembling, and link-editing an HLASM routine

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 OS/390 or TSO, or CMS.

Writing an edit routine for native OS/390, TSO, or ISPF without LE

The QMF edit exit interface for PL/I in TSO, ISPF, or native OS/390 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 167 on page 525 shows the program structure of a PL/I edit exit routine in TSO, ISPF, or native OS/390.

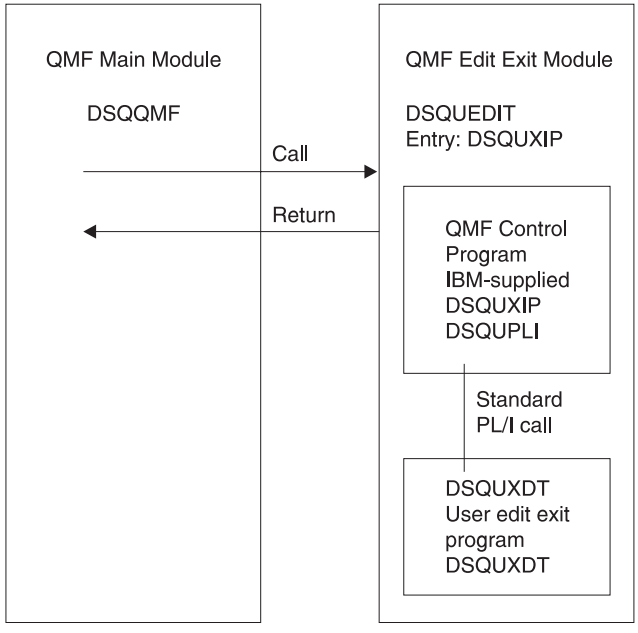


Figure 166. Program structure of a PL/I edit exit routine without LE

How a PL/I edit routine interacts with native OS/390, 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 QMF720.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 QMF720.SDSQUSRE on OS/390 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 QMF720.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 native OS/390.

```
//samPLI     JOB
//STEP1      EXEC IEL1CL
//* Provide Access to QMF Edit Macro DXEECSP
//PLI.SYSLIB   DD  DSN=QMF720.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 QMF720.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 on VM without LE

The QMF edit exit interface for PL/I in CMS 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 167 shows the program structure of a PL/I edit exit routine in CMS.

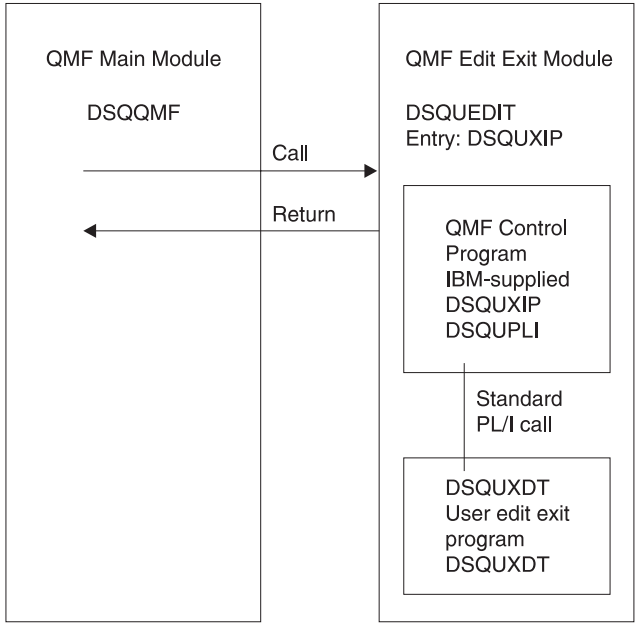


Figure 167. Program structure of a PL/I edit exit routine without LE

Example program DSQUXDTP

The IBM-supplied example edit exit program in PL/I, named DSQUXDTP, is located on the QMF production disk on CMS. 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. If you build your own routine instead, note that within the source is an %INCLUDE statement for DXEECSP, which is a member of DSQUSERE MACLIB on CMS. It is DXEECSP that defines the input fields, giving them the names we are using in this chapter. It is best to include this in your own edit routine.

How a PL/I edit routine interacts with QMF

Linkage begins with the PROCEDURE statement:

```
DSQUXDT:
        PROCEDURE(DXEECSF,ECSINPTF,ECSRSLTF) ...;
```

Passed through this statement are the control block (DXEECSF), the value to be formatted (ECSINPTF), and the storage set aside for the formatted result (ECSRSLTF). At this point, you can expect to find declarations defining DXEECSF as a structure, and defining ECSINPTF and ECSRSLTF as character strings. Instead, you find the statement:

```
DECLARE (DXEECSF,
         ECSINPTF,
         ECSRSLTF)
            BINARY FIXED, ...
```

which defines the three parameters as fullword integers. This is because the calling program itself, in order to avoid the overhead of locators and descriptors, represents the parameters in its call to DSQUXDT as fullword integers. QMF doesn't know in what language the calling program is written, so the parameters are passed in the same way as they are for Assembly.

In the sample program below, the actual parameter descriptions appear in the previously mentioned block of definitions comprising DXEECSP. The declaration for the control block begins with:

```
DECLARE
  1 DXEECSP BASED(ECSPTR)
.
.
.
```

The statements defining the other two parameters are:

```
DECLARE
  ECSINPT CHARACTER(32767)
          BASED(ECSINPTP), ... and
DECLARE
  ECSRSLT CHARACTER(32767)
          BASED(ECSRSLTP);
```

Thus, the parameters are defined as based storage. To complete the linkage, the pointers are set to the appropriate addresses at the start of the procedural logic section:

```
ECSPTR   = ADDR(DXEECSF);
ECSINPTP = ADDR(ECSINPTF);
ECSRSLTP = ADDR(ECSRSLTF);
```

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.

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 DSQUSERE MACLIB on the QMF production disk on CMS must be available in a macro library. You need to make the macro libraries available to the PL/I compiler by issuing a CMS GLOBAL MACLIB command. For example:

```
GLOBAL MACLIB DSQUSERE PLICOMP
```

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.

Creating your DSQUEDIT module file in PL/I

Before you can create your DSQUEDIT module file, ensure that you can access the IBM-supplied control module (DSQUXIP). DSQUXIP is located on the QMF production disk. You need to access this disk prior to creating the module file.

To create the DSQUEDIT module file, use the CMS LOAD and GENMOD commands:

1. Load the text files that make up the DSQUEDIT module.

   The DSQUEDIT module must be relocatable. To be relocatable, the module must be loaded with RLD entries. You do this by specifying the RLDSAVE option on the CMS LOAD command. The entry point to the DSQUEDIT module must be DSQUXIP. PL/I text libraries must be made available by issuing a CMS GLOBAL TXTLIB command. Issue the following CMS commands:

```
GLOBAL TXTLIB IBMLIB PLILIB
LOAD DSQUXIP DSQUXDT DSQUPLI (RLDSAVE RESET DSQUXIP)
```

You can run your edit routine in either 24-bit or 31-bit addressing mode. QMF manages address switching as required. You can specify 31-bit addressing on the CMS LOAD command. For example:

```
GLOBAL TXTLIB IBMLIB PLILIB
LOAD DSQUXIP DSQUXDT DSQUPLI
     (RLDSAVE RESET DSQUXIP AMODE 31 RMODE ANY)
```

2. Generate the DSQUEDIT module.

   Issue the CMS GENMOD command to generate the DSQUEDIT module from the text files just loaded by the CMS LOAD command:

   ```
   GENMOD DSQUEDIT
   ```

Once the user edit routine is tested, it can replace the DSQUEDIT module file on the QMF production disk or user disk that is available when starting QMF. In order to use the PL/I user edit routine, the PL/I production disk and run-time libraries need to be available when you start QMF.

When running under ISPF and starting QMF using the PGM form of ISPSTART, the PL/I run-time load libraries must be specified using a CMS FILEDEF command for ISPLLIB. For guidelines and considerations about PL/I programs running in ISPF, see *ISPF for VM Dialog Management Services and Examples*

When running without ISPF, or running under ISPF and starting QMF using the program segment form of ISPSTART, the PL/I run-time load libraries must be specified using a CMS GLOBAL LOADLIB command.

For detailed information on how to compile and make run-time libraries available for PL/I, see *PL/I Programming Guide.*

Writing an edit routine in PL/I with language environment (LE)

Use these instructions for writing an edit routine for native OS/390, TSO, or CMS with language environment.

Writing an edit routine in PL/I for native OS/390, TSO, or ISPF with language environment (LE)

The QMF edit exit interface for PL/I in TSO, ISPF, or native OS/390 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 168 shows the program structure of a PL/I edit exit routine in TSO, ISPF, or native OS/390.

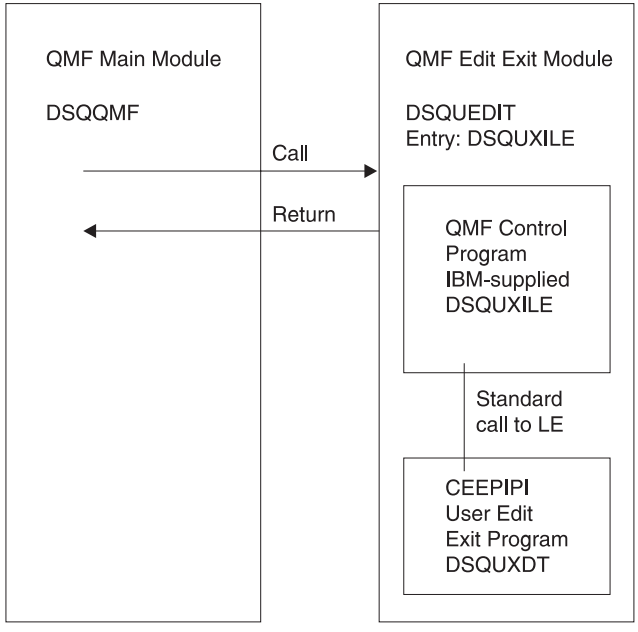


Figure 168. Program structure of a PL/I edit exit routine with LE

How a PL/I Edit routine interacts with native OS/390, 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 QMF720.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 QMF720.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 native OS/390.

```
 //samPLI     JOB
 //STEP1      EXEC PLIXCL
 //* Provide Access to QMF Edit Macro DXEECSP
 //PLI.SYSLIB   DD  DSN=QMF720.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=QMF720.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 QMF720.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 VM with language environment (LE)

The QMF edit exit interface for PL/I in CMS 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 169 shows the program structure of a PL/I edit exit routine in CMS.

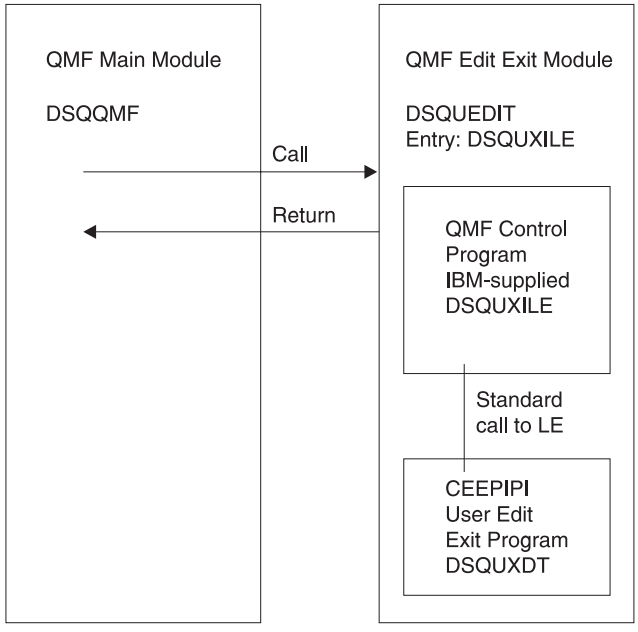


Figure 169. Program structure of a PL/I edit exit routine with LE

Generating your PL/I program for LE

Before you can create your DSQUEDIT module file, ensure that you can access the IBM-supplied module DSQUXILE. This module is located on the production disk. You need to access this disk prior to creating the module file. Use the CMS LOAD and GEMOD commands as follows:

1. Load the text files that make up the DSQUEDIT module.

   The DSQUEDIT module must be relocatable. To be relocatable, the module must be loaded with RLD entries. Do this by specifying the RLDSAVE option on the CMS/LOAD command. The entry point to the DSQUEDIT module must be DSQUXILE. LE text libraries must be made available by using a CMS GLOBAL TXTLIB command:

   ```
   GLOBAL TXTLIB SCEELKED
   LOAD DSQUXILE DSQUXDT ( RLDSAVE RESET DSQUXILE
   ```

You can run your edit routine in either 24-bit or 31-bit addressing mode. QMF manages address switching as required. You can specify 31-bit addressing on the CMS LOAD command:

```
GLOBAL TXTLIB SCEELKED
LOAD DSQUXILE DSQUXDT ( RLDSAVE RESET DSQUXILE AMODE 31 RMODE ANY
```

2. Generate the DSQUEDIT module. Issue the CMS GENMOD command to generate the DSQUEDIT module from the text files just loaded by the CMS LOAD command:

```
GENMOD DSQUEDIT
```

An example procedure statement specifying parameters is as follows:

```
DSQUXDT:
    PROCEDURE(DXEECSF,ECSINPTF,ECSRSLTF) OPTIONS(REENTRANT);
```

Writing an edit routine in PL/I for CICS on OS/390

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 170 shows the program structure of a PL/I edit exit routine in CICS.

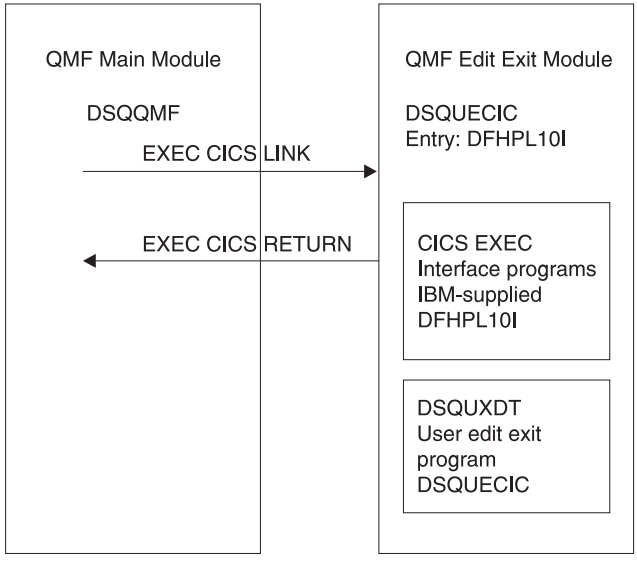


Figure 170. Program structure for PL/I edit exit routine in CICS

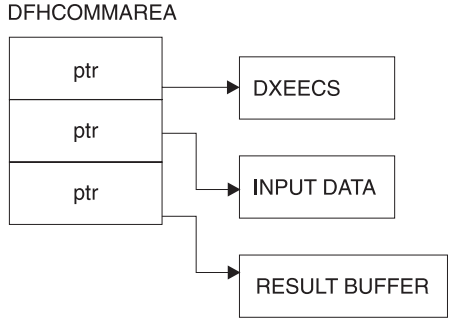
Example program DSQUXCTP

The IBM-supplied example edit program in Assembly, named DSQUXCTP, is located in QMF sample library QMF720.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 QMF720.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.



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 QMF720.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 OS/390

QMF edit exit interface control block DXEECSP, located in QMF sample library QMF720.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 OS/390

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='QMF720.SDSQLOAD'
//TRN.SYSIN DD *
      .
      Your program or modified copy of QMF sample DSQUXCTP
      .
/*
//* Provide access to QMF Edit Macro DXEECSP
//PLI.SYSLIB DD  DSN=QMF720.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 Assembly. 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 PL/I for CICS/VSE

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 DFHPL1I.
- Your edit exit program, which is named DSQUECIC

Figure 171 on page 536 shows the program structure of a PL/I edit exit routine in CICS.

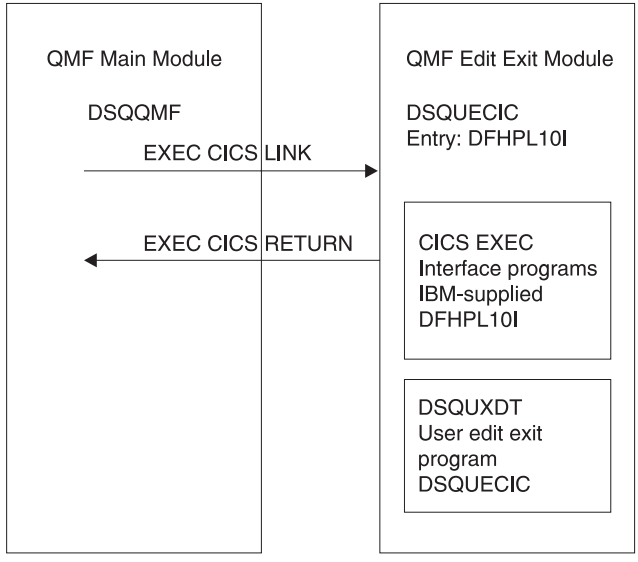


Figure 171. Program structure for PL/I edit exit routine in CICS

Example program DSQUXCTP

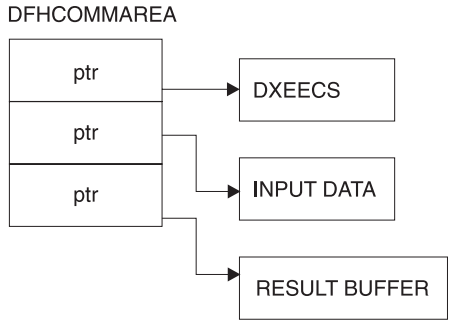
The IBM-supplied example edit program in Assembly, named DSQUXCTP, is located in QMF sublibrary on VSE as DSQUXCTP.Z. The example program is heavily commented; you can print it, browse it online, or modify it to meet your needs. A PL/I data structure is shipped with QMF as DXEECSP, located in DXEECSP.C on VSE in the QMF sublibrary. 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.



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 QMF720.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

On VSE, before you translate your program, include in the LIBDEF statement the QMF edit exit interface control block DXEECSP.C, which is located in the sublibrary where QMF is installed.

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.

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 VSE

The sample job DSQ3XCTP.Z is shipped with QMF. This job translates, compiles, and link-edits the example PL/I program (DSQUXCTP.Z), which is also shipped with QMF. Use the sample job as a starting point to create JCL that translates, assembles, and link-edits your own edit exit routine.

Ignore weak external references unresolved by the linkage editor, and also the associated messages about unresolved address constants. For more information on installing a program in CICS, see the CICS System Definition Guide.

```
..* $$ JOB JNM=DSQ3XCTP,DISP=D,CLASS=0
// JOB DSQ3XCTP  Sample Job to Install QMF Edit Exit for PL/I
* ------------------------------------------------------------------
* Install QMF edit exit (PL/I)
* ------------------------------------------------------------------
// SETPARM VOLID=volid      *-- update volid for syspch
// SETPARM START=rtrk       *-- update start track/block
// SETPARM SIZE=ntrks       *-- update number of tracks/blocks
* ------------------------------------------------------------------
// DLBL   IJSYSPH,'CICS.TRANSLAT.OUTPUT',0
// EXTENT SYSPCH,,1,0,&START,&SIZE
ASSGN SYSPCH,DISK,VOL=&VOLID,SHR
*  Library search chain must contain the QMF, CICS and PL/I sublibrary
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE,PRD2.CONFIG)
// LIBDEF PHASE,CATALOG=PRD2.PROD
* ------------------------------------------------------------------
* Step 1: Translate user edit exit program
* ------------------------------------------------------------------
// EXEC  DFHECP1$,SIZE=256K,PARM='XOPTS(CICS,QUOTE)'
..* $$ SLI MEM=DSQUXCTP.Z,S=PRD2.PROD
/*
* ------------------------------------------------------------------
* Step 2: Compile  translated user edit exit program
* ------------------------------------------------------------------
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'CICS.TRANSLAT.OUTPUT',0
// EXTENT SYSIPT
ASGN SYSIPT,DISK,VOL=&VOLID,SHR
// OPTION CATAL
   PHASE DSQUECIC,*,SVA
         INCLUDE DFHPLII
// EXEC PLIOPT
CLOSE SYSIPT,SYSRDR
/*
* ------------------------------------------------------------------
* Step 3: Link-edit user edit exit program
* ------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=31'
/*
/&
// JOB RESET
ASSGN SYSIPT,SYSRDR   IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D      IF 1A93D, CLOSE SYSPCH,00D
/&
..* $$ EOJ
```

Figure 172. Example JCL for translating, assembling and link-editing an HLASM routine

How a PL/I program interacts with QMF

The interface control block between QMF and the user edit interface DSQUECIC is DCXEECS. 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 edit routine.

Writing an edit routine in COBOL without language environment (LE)

You can write an edit routine in COBOL for native OS/390,, TSO, or CMS.

In this section, COBOL refers to VS COBOL II, COBOL/370, and COBOL for OS/390 and VM unless otherwise stated.

Writing an edit routine in COBOL for native OS/390, 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 173 shows the program structure of a COBOL edit exit routine

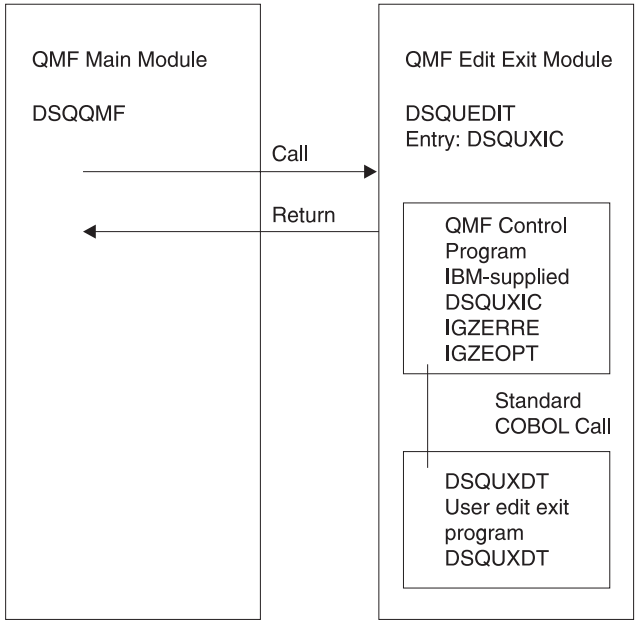


Figure 173. Program structure of a COBOL edit exit routine

Example program DSQUXDTC

The IBM-supplied example edit exit program in COBOL, named DSQUXDTC, is located in QMF sample library QMF720.SDSQSAPE on OS/390 . 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 QMF720.SDSQUSRE on OS/390.

Select COBOL compiler options as follows:

COBOL II

Specify compiler options RENT, RES, NODYNAM, OBJECT, and LIB.

COBOL/370 or **IBM COBOL for OS/390 and VM**

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 QMF720.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 Assembly 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 OS/390

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 QMF720.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 OS/390

The following are example statements for compiling and link-editing your job for TSO or native OS/390).

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=QMF720.SDSQUSRE,DISP=SHR
//COB2.SYSIN    DD  *
          .
          Your program or copy of QMF sample DSQUXDTC
          .
/*
```

```
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QMF720.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 OS/390:

```
//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=QMF720.SDSQUSRE,DISP=SHR
//COBOL.SYSIN   DD  *
          .
          Your program or copy of QMF sample DSQUXDTC
          .
/*
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QMF720.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 for CMS 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
- Control macro, which is supplied by IBM as IGZOPT
- Control module, which is supplied by IBM as IGZERRE
- Your edit exit program, which is named DSQUXDT

Figure 174 shows the program structure of a COBOL edit exit routine

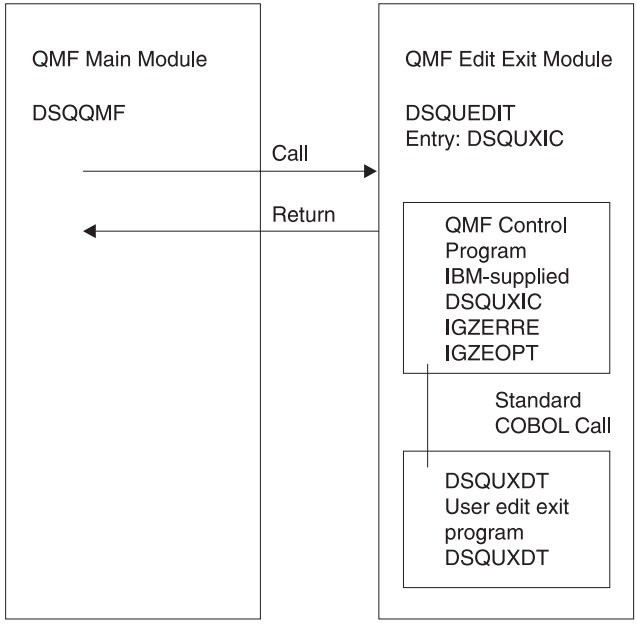


Figure 174. Program structure of a COBOL edit exit routine

Example program DSQUXDTC

The IBM-supplied example edit exit program in COBOL, named DSQUXDTC, is located in the QMF production disk on CMS. 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. If you plan to write your own user edit routine, note that this routine contains a COPY statement for DXEECSC, which is a member of DSQUSERE MACLIB on CMS. It is DXEECSC that defines the input fields, giving them the names we are using in this chapter. It is best to include this in your own user edit routine.

How a COBOL edit routine interacts with QMF

The following statement begins the mainline logic:

```
PROCEDURE DIVISION USING DXEECS, ECSINPT, ECSRSLT
```

In this example, DXEECS is the name of the control block, ECSINPT is the name of the value to be formatted, and ECSRSLT is the name of the area reserved for the formatted result. The fields within these parameters are defined in DXEECSC.

The interface control block between QMF and the user edit interface DSQUXDT is DXEECS. It contains the user's edit code 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 edit routine. Return control to QMF with a GOBACK statement.

Please refer to the DXEECSC file provided by QMF as a sample COBOL version of the DXEECS control block. This file is located in library QMF720.SDSQSAPE on OS/390, or on the QMF production disk on CMS.

Compiling your program

Compile DSQUXDT (the edit exit program you have written). During the compile, QMF edit exit interface control block DXEECSC, located in in the DSQUSERE MACLIB on the QMF production disk on CMS, must be available in a macro library.

On CMS, you need to access the QMF and COBOL production disks. You also need to make the macro libraries available to the COBOL compiler by issuing a CMS GLOBAL MACLIB command. For example:

```
GLOBAL MACLIB DSQUSERE VSC2MAC
```

DXEECSC, as distributed by IBM, uses quotation marks ("") to delimit character literals. If your program uses apostrophes ('), you must either change DXEECSC as distributed by IBM or copy the structure to your program, changing quotes to apostrophes.

Select COBOL compiler options as follows:

COBOL II

Specify compiler options RENT, RES, NODYNAM, OBJECT, and LIB.

COBOL/370 or **IBM COBOL for OS/390 and VM**

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.

Assembling the run time options module

On CMS, use the C2CUSTL EXEC provided by IBM to assemble IGZOPT. Follow the prompts and add option STAE=NO to the IGZEOPT ASSEMBLE file. The new or changed option file is replaced in VSC2LTXT TXTLIB and VSC2LOAD LOADLIB, or in another TXTLIB and LOADLIB that you specify. Refer to VS COBOL II Installation and Customization for CMS for more information about assembling run time options.

Generating your program on CMS

Before you can create the module file, ensure that you can access the IBM-supplied control module (DSQUXIC). DSQUXIC is located on the QMF production disk. You need to access this disk prior to creating the module file.

To create the DSQUEDIT module file, use the CMS LOAD and GENMOD commands as follows:

1. Load the text files that make up the DSQUEDIT module.

   The DSQUEDIT module must be relocatable. To be relocatable, the module must be loaded with RLD entries. You do this by specifying the RLDSAVE option on the CMS LOAD command. The entry point to the DSQUEDIT module must be DSQUXIC. COBOL text libraries must be made available by issuing a CMS GLOBAL TEXTLIB command. Issue the following CMS commands:

   ```
   GLOBAL TXTLIB VSC2LTXT
   LOAD DSQUXIC DSQUXDT (RLDSAVE RESET DSQUXIC)
   ```

   You can run your edit routine in either 24-bit or 31-bit addressing mode. QMF manages address switching as required. You can specify 31-bit addressing on the CMS LOAD command. For example:

   ```
   GLOBAL TXTLIB VSC2LTXT
   LOAD DSQUXIC DSQUXDT
        (RLDSAVE RESET DSQUXIC AMODE 31 RMODE ANY)
   ```

2. Issue the CMS GENMOD command to generate the DSQUEDIT module from the text files just loaded by the CMS LOAD command:

   ```
   GENMOD DSQUEDIT (AMODE 31 RMODE ANY)
   ```

Once the user edit routine is tested, it can replace the DSQUEDIT module file on the QMF production disk or user disk that is available when you start QMF. In order to use the COBOL user edit routine, the COBOL production disk and run-time libraries need to be available when you start QMF.

When running under ISPF and starting QMF using the PGM form of ISPSTART, the COBOL run-time load libraries must be specified using a CMS FILEDEF command for ISPLLIB. For guidelines and considerations about COBOL programs running in ISPF, see the *ISPF for VM Dialog Management Services and Examples* manual.

When running without ISPF, or running under ISPF and starting QMF using the program segment form of ISPSTART, the COBOL run-time load libraries must be specified using a CMS GLOBAL LOADLIB command.

For detailed information on how to compile and make run-time libraries available for COBOL, see *VS COBOL II Application Programming Guide*.

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 OS/390, TSO, and CMS.

Writing an edit routine in COBOL for native OS/390, ISPF, and TSO with language environment (LE)

The QMF edit exit interface of COBOL in native OS/390 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 175 on page 548 shows the program structure of a COBOL edit exit routine in native OS/390 and TSO.

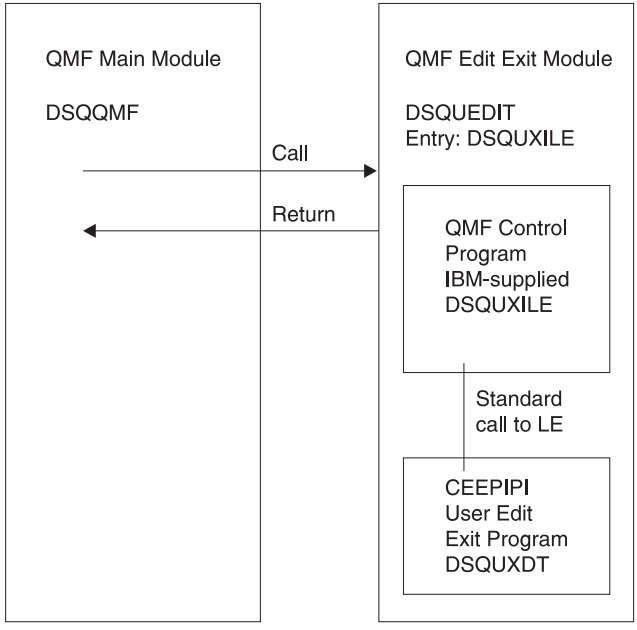


Figure 175. Program structure of a COBOL edit exit routine in TSO, ISPF, or native OS/390 with LE

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 QMF720.SDSQSAPE on OS/390. 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 OS/390, 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 QMF720.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 QMF720.SDSQUSRE on OS/390 or on the QMF production disk on CMS, 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 QMF720.SDSQLOAD on OS/390).

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 OS/390

The following are example statements for compiling and link-editing your job for TSO or native OS/390:

```
//samCOBOL   JOB
//STEP1     EXEC PROC=IGYWCL
//* Provide Access to QMF Edit Macro DXEECSC
//COBOL.SYSLIB  DD  DSN=QMF720.SDSQUSRE,DISP=SHR
//COBOL.SYSIN   DD  *
```

Your program or copy of QMF sample DSQUXDTC:

```
/*
//* Provide Access to QMF Interface Module
//LKED.QMFLOAD  DD DSN=QMF720.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 CMS with language environment (LE)

The QMF edit exit interface of COBOL in CMS 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 176 shows the program structure of a COBOL edit exit routine in CMS.

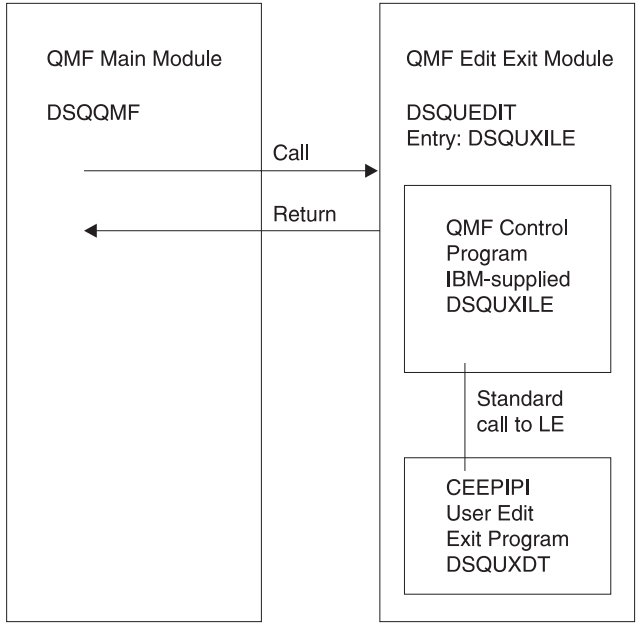


Figure 176. Program structure of a COBOL edit exit routine in CMS with LE

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.

Generating your COBOL program for LE in CMS

Before you can create the module file, ensure that you can access the IBM-supplied control module (DSQUXILE). DSQUXILE is located on the QMF production disk. You need to access this disk prior to creating the module file.

To create the DSQUEDIT module file , use the CMS LOAD and GENMOD commands as follows:

1. Load the text files that make up the DSQUEDIT module. The DSQUEDIT module must be relocatable To be relocatable, the module must be loaded with RLD entries. You do this by specifying the RLDSAVE option on the CMS/LOAD command. The entry point to the DSQUEDIT module must be DSQUXILE. LE text libraries must be made available by issuing a CMS GLOBAL TXTLIB command. Issue the following CMS command:

   ```
   GLOBAL TXTLIB SCEELKED
   LOAD DSQUXILE DSQUXDT ( RLDSAVE RESET DSQUXILE
   ```

   You can run your edit routine in either 24-bit or 31-bit addressing mode. QMF manages address switching as required. You can specify 31-bit addressing on the CMS LOAD command. For example:

   ```
   GLOBAL TXTLIB SCEELKED
   LOAD DSQUXILE DSQUXDT ( RLDSAVE RESET DSQUXILE AMODE 31 RMODE ANY
   ```

2. Generate the DSQUEDIT module.

   Issue the CMS GENMOD command to generate the DSQUEDIT module from the text files just loaded by the CMS LOAD command:

   ```
   GENMOD DSQUEDIT
   ```

Writing an edit routine in COBOL for CICS on OS/390

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 177 on page 552 shows the structure of a COBOL edit exit routine in CICS.

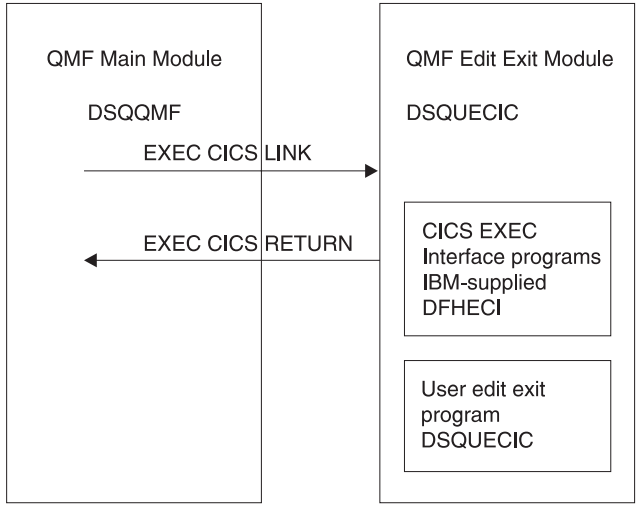
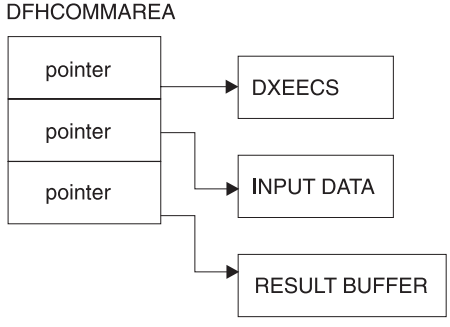


Figure 177. Program structure for a COBOL edit exit routine in CICS

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 QMF720.SDSQSAPE on OS/390. 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 QMF720.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 OS/390

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='QMF720.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=QMF720.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 OS/390

When QMF is installed, the QMF edit exit program is installed with a program language of Assembly. 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 QMF720.SDSQSAPE on OS/390. 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.

Writing an edit routine in COBOL for CICS/VSE

The edit exit interface for COBOL in CICS consists of these parts:

- Interface control block, which is shipped with QMF as DXEECSC.C
- CICS command interface module, which is shipped with CICS as DFHECI
- Your edit exit program, which is named DSQUECIC

Figure 178 shows the structure of a COBOL edit exit routine in CICS.

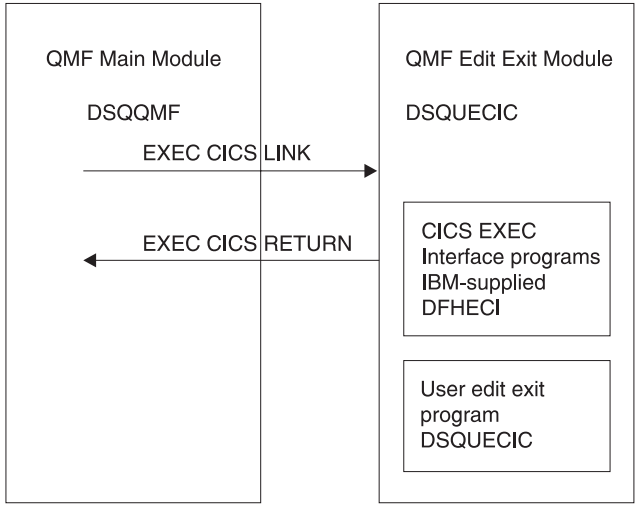


Figure 178. Program structure for a COBOL edit exit routine in CICS/VSE

Example program DSQUCTC

The IBM-supplied example edit program in COBOL named DSQUXCTC.Z is located in the QMF sublibrary on VSE. The example program is heavily

commented; it can be browsed online, printed, or modified to suit your needs. A sample job named DSQ3XCTC.Z is shipped with QMF. This job compiles and link-edits the sample COBOL program as DXEECSC, located in library QMF720.SDSQUSRE on OS/390 or as DXQUCTC.Z .

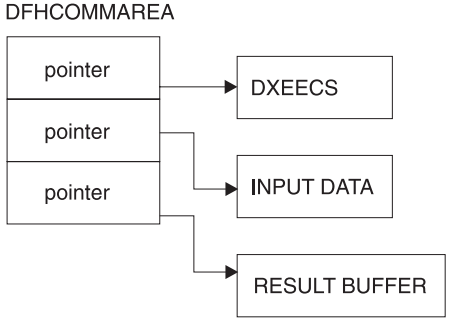
Literal delimiters: quotes or apostrophes

You must use either quotes (") or apostrophes (') to delimit literals in a COBOL program. You can specify the delimiter of your choice to the CICS translation process and the COBOL compiler by specifying "QUOTE" or "APOST". Make sure the APOST or QUOTE option in effect for the COBOL compiler matches that of the CICS translator.

The edit control block DXEECSC.C and the sample COBOL program DSQUXCTC.Z, 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.

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 QMF720.SDSQSAPE on OS/390 or as DXEECSC.C in the QMF sublibrary on VSE. 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.
```

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.

Translating your COBOL program

Before you translate your program, include the QMF edit exit interface control block, DXEECSC.C, in the LIBDEF statement. DXEECSC.C is located in the sublibrary where QMF is installed.

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

Specify COBOL compiler options RENT, RES, and NODYNAM, and run time options NOSTAE and NORTEREUS.

QMF distributes the user edit routine control block DXEECSC.C, 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 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. DSQUECIC must be executable in 31-bit addressing mode.

Example JCL statements for translating, compiling, and link-editing on VSE

Figure 179 on page 559 shows the sample job DSQ3XCTC.Z, which is shipped with QMF. This job translates, compiles, and link-edits the example COBOL program (DSQUXCTC.Z), which is also shipped with QMF. Use the sample job as a starting point to create JCL that translates, assembles, and link-edits your own edit exit routine.

Ignore weak external references unresolved by the linkage editor, and also the associated messages about unresolved address constants. For more information on installing a program in CICS, see CICS System Definition Guide.

```
// JOB DSQ3XCTC  Install QMF Edit Exit for COBOL
* ------------------------------------------------------------------
* Install QMF edit exit (COBOL Version)
* ------------------------------------------------------------------
// SETPARM VOLID=volid         *-- update volid for syspch
// SETPARM START=rtrk          *-- update start track/block
// SETPARM SIZE=ntrks          *-- update number of tracks/blocks
* ------------------------------------------------------------------
// DLBL   IJSYSPH,'CICS.TRANSLAT.OUTPUT',0
// EXTENT SYSPCH,,1,0,&STARTL,&SIZE.
ASSGN SYSPCH,DISK,VOL=&VOLID.,SHR
*  Library search chain must contain the QMF, CICS and COBOL sublibraries
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE,PRD2.CONFIG)
// LIBDEF PHASE,CATALOG=PRD2.PROD
* ------------------------------------------------------------------
* Step 1: Translate user edit exit program
*-------------------------------------------------------------------
// EXEC   DFHECP1$,SIZE=256K,PARM='XOPTS(CICS,QUOTE)'
  :
 COBOL source program here
  :
/*
*-------------------------------------------------------------------
* Step 2: Compile  translated user edit exit program
*-------------------------------------------------------------------
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'CICS.RANSLAT.OUTPUT',0
// EXTENT SYSIPT
ASSIGN SYSIPT,DISK,VOL=&VOLID.,SHR
// OPTION CATAL
   PHASE DSQUECIC,*,SVA
         INCLUDE DFHEC1
// EXEC IGYCRCTL,PARM='SZ(MAX),OBJECT,MAP,RES,NODYNAM,QUOTE,LIB,RENT
CLOSE SYSIPT,SYSRDR
/*
*-------------------------------------------------------------------
* Step 3: Link-edit user edit exit program
*-------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=31'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR   IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D      IF 1A93D, CLOSE SYSPCH,00D
/&
```

Figure 179. Example JCL for translating, compiling, and link-editing a COBOL routine

Defining the edit exit phase to CICS on VSE

During QMF installation, the QMF edit exit program is installed with a programming language of HLASM. To use the COBOL edit exit program, you must define the routine to CICS using the COBOL

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 30. 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 the number of rows a user can retrieve from the database. You can use this default exit routine, or use Assembler to modify the routine or write one of your own.

Using a governor exit routine on OS/390

On OS/390, 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 OS/390 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 the *QMF High Performance Option User's Guide for OS/390*.

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 OS/390 (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 180. Message displayed when a resource limit is approaching. The n symbol in the figure represents an NLID from Table 1 on page xiv

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 180. 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 181. 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 181. 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 182. 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 OS/390 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 183. Message displayed when a resource limit is approaching (OS/390). The n symbol in the figure represents an NLID from Table 1 on page xiv

- 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 184 from the SQL query panel.

```
UPDATE Q.RESOURCE_VIEW
SET INTVAL=0
WHERE RESOURCE_OPTION='SCOPE' AND
      RESOURCE_GROUP='SYSTEM'
```

Figure 184. 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 185 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 185. 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 566. Then see the procedure in "Defining your own resource limits" on page 569.

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 70 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 OS/390 batch. Use the CHARVAL column to indicate the limits defined in each row, as shown.

Table 70. 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 71. Options for the IBM-supplied governor exit for TSO, ISPF, or native OS/390 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 minutes interval between time check

Table 72. Options for the IBM-supplied governor exit for CMS

GROUP	OPTION	INTVAL	FLOATVAL	CHARVAL
SYSTEM	TIMELIMIT	3600	-	Cancel after 60 minutes
SYSTEM	TIMEPROMPT	900	-	Prompt user after 15 minutes
SYSTEM	TIMECHECK	900	-	15 minutes 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 OS/390 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 186 shows how the governor limits use of resources.

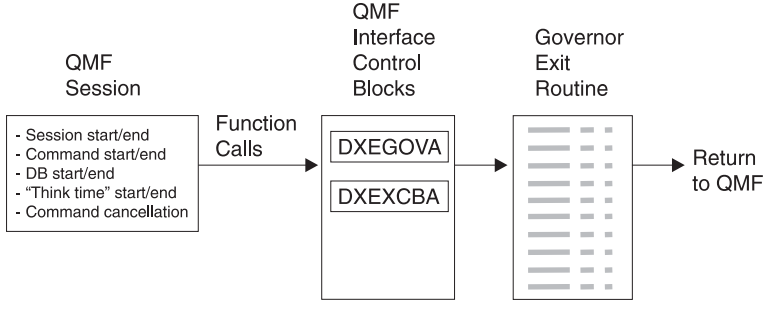


Figure 186. How a governor exit routine works with QMF for OS/390

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 599.

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 181 on page 564. 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 591 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 OS/390 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 187 on page 570 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 on page 570

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
    VALUES('GROUP1','ROWPROMPT',10000)
```

Figure 187. Activating prompting for row limit

2. Run the query in Figure 188 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 188. Activating cancellation of activities when user reaches row limit

3. Run the query in Figure 189 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 189. Activating prompting for time limit

4. Run the query in Figure 190 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 190. For TSO and native OS/390 batch: Activating cancellation of activities when user reaches time limit

5. Run the query in Figure 191 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 191. For TSO and native OS/390 batch: Activating time interval check

6. Run the query shown in Figure 192 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 192. Turning on the governor for a particular resource group

7. Run a query similar to the one in Figure 193 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 193. 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 xiv.
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 73 on page 573.

Figure 194 on page 572 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 194 and Figure 195. 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 194. Creating a resource control table or renaming Q.RESOURCE_TABLE

When running QMF on OS/390, 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 OS390 SQL Reference* manual.

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 195 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 195. 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 73. 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.
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.

Using a governor exit routine on VM

Default limits are provided for the amount of time spent running a QMF command.

Using the IBM-supplied governor exit routine

The governor exit routine supplied for CMS (DSQUEGV2) controls how many rows a user can retrieve from the database or the real 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 196. Message displayed when a resource limit is approaching. The n symbol in the figure represents an NLID from Table 1 on page xiv

- 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 196. 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 197. 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 197. 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 198. 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.

If you want to define your own limits for when the user is warned and when data retrieval is canceled, see “Defining your own resource limits” on page 569.

CMS has two 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 real time for the cycle has reached 15 minutes, at which time the user sees the message shown in Figure 199 on page 575.
- A time limit value cancels the command when 60 minutes of real time are used during the cycle.

```
DSQUn00 QMF governor prompt:
  Command has executed for 15 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 199. Message displayed when a resource limit is approaching

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 below from the SQL query panel.

```
UPDATE Q.RESOURCE_VIEW
SET INTVAL=0
WHERE RESOURCE_OPTION='SCOPE' AND
      RESOURCE_GROUP='SYSTEM'
```

Figure 200. 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 below activate default values for user JONES (using English QMF) and user SCHMIDT (using German QMF).

```
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 201. Updating a user's resource group

3. Instruct users to reconnect to the database to activate the new values.

"How a governor exit routine controls resources" on page 566 explains how the governor uses the information in the Q.RESOURCE__VIEW and the Q.PROFILES table to control resources.

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 566. Then see the procedure in "Defining your own resource limits" on page 569.

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 74 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.

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, as shown in Table 74. The group has additional time options for TSO, native OS/390 batch, or CMS. Use the CHARVAL column to indicate the limits defined in each row, as shown.

Table 74. 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 75. Options for the IBM-supplied governor exit for TSO or native OS/390 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 minutes interval between time check

Table 76. Options for the IBM-supplied governor exit for CMS

GROUP	OPTION	INTVAL	FLOATVAL	CHARVAL
SYSTEM	TIMELIMIT	3600	-	Cancel after 60 minutes
SYSTEM	TIMEPROMPT	900	-	Prompt user after 15 minutes
SYSTEM	TIMECHECK	900	-	15 minutes 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.

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.

The three additional options provided in CMS are:

TIMELIMIT = 3600
If the user decides to continue when warned, the governor exit routine cancels the command after 60 minutes of processor time have

elapsed. TIMELIMIT is checked at TIMECHECK intervals; therefore, more than 60 minutes of processor time can elapse if the TIMECHECK interval is set at an interval not divisible by 60.

TIMEPROMPT = 900
Warns the user when 15 minutes of processor time have elapsed.

TIMECHECK = 900
Specifies 15 minutes of real time between time checks or prompting or canceling.

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 202 shows how the governor limits use of resources.

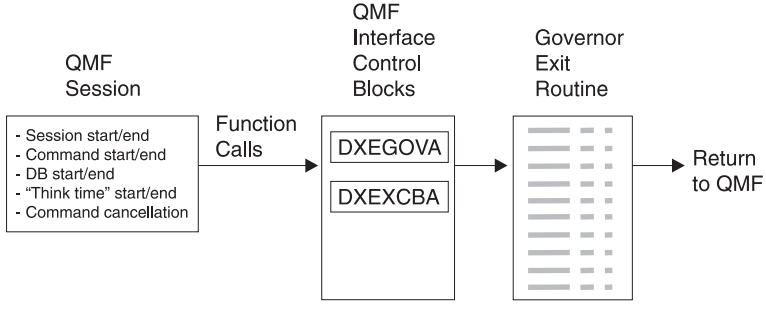


Figure 202. How a governor exit routine works with QMF CMS

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 599.

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 181 on page 564. See "Defining your own resource limits" on page 569 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 591 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 CMS, the governor also prompts a user in GROUP1 when the real time reaches 10 minutes, and cancels the user's activity when the real time reaches 45 minutes. The procedure also shows an example of how to add a user to a resource group.

1. Run the query below 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 203. Activating prompting for row limit

2. Run the query in Figure 204 on page 580 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 204. Activating cancellation of activities when user reaches row limit

3. Run the query in Figure 205 to set the real 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',600)
```

Figure 205. Activating prompting for time limit on VM

4. Run the query in Figure 206 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',2700)
```

Figure 206. For CMS: Activating cancellation of activities when user reaches time limit

5. Run the query in Figure 207 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',600)
```

Figure 207. For CMS: Activating time interval check

6. Run the query shown in Figure 208 on page 581 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 208. Turning on the governor for a particular resource group

7. Run a query similar to the one in Figure 209 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 209. 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 xiv.
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 in Table 77 on page 582.

Figure 210 on page 582 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 210 on page 582 and Figure 211 on page 582. Substitute your own dbspace 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 DBSPACE1
```

Figure 210. Creating a resource control table or renaming Q.RESOURCE_TABLE on VM

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 211 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 211. Redefining the Q.RESOURCE_VIEW

Table 77. 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.
FLOATVAL	FLOAT		Yes	Reflects resource limit for resource options that have floating point values. FLOATVAL is null for the IBM-supplied governor.

Table 77. Structure of the Q.RESOURCE_TABLE table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
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.

Using a governor exit routine on VSE

Use these instructions to control QMF resources on VSE.

Using the IBM-supplied governor exit routine

The governor exit routine supplied by IBM controls how many rows a user can retrieve from the database. 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 in Figure 212.

```
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 212. Message displayed when a resource limit is approaching. The n symbol in the figure represents an NLID from Table 1 on page xiv

- 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 212. When the IBM-supplied governor cancels data retrieval, the user sees the message shown in Figure 213.

```
Row limit exceeded!  Your command canceled by QMF governor.
```

Figure 213. 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 213. For example, if your

procedure contains a command that requires the report to complete (such as ERASE), you receive the message shown in Figure 214.

```
Procedure canceled.
```

Figure 214. 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.

If you want to define your own limits for when the user is warned and when data retrieval is canceled, see "Defining your own resource limits" on page 579.

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 below from the SQL query panel.

```
UPDATE Q.RESOURCE_VIEW
SET INTVAL=0
WHERE RESOURCE_OPTION='SCOPE' AND
      RESOURCE_GROUP='SYSTEM'
```

Figure 215. 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 216 on page 585 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 216. Updating a user's resource group

3. Instruct users to reconnect to the database to activate the new values.

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 576. Then see the procedure in "Defining your own resource limits" on page 588.

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 78 on page 586 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.

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. Use the CHARVAL column to indicate the limits defined in each row, as shown.

Table 78. 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

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.

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.

QMF calls the governor exit routine at a number of different points within the QMF session, as shown in Figure 217.

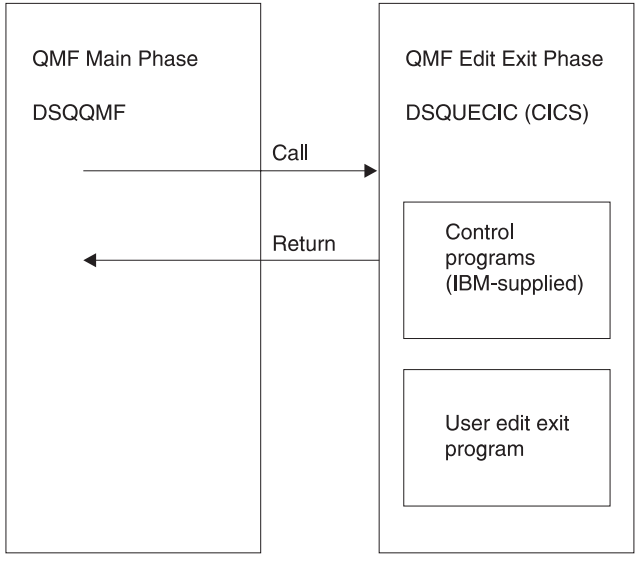


Figure 217. How a governor exit routine works with QMF for VSE

These calls are called function calls. For more information about function calls, see "Points at which QMF calls the governor" on page 599.

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 213 on page 583. See "Defining your own resource limits" on page 588 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 591 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. The procedure also shows an example of how to add a user to a resource group.

1. Run the query below 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 218. Activating prompting for row limit

2. Run the query in Figure 219 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 219. Activating cancellation of activities when user reaches row limit

3. Run the query shown in below 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 220. Turning on the governor for a particular resource group

4. Run a query similar to the one below 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 221. 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 xiv.
5. Instruct the user whose profile you updated to end the current QMF session and start 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 in Table 79 on page 590.

Figure 222 on page 590 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 222 on page 590 and Figure 223 on page 590. Substitute your own dbspace 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 DBSPACE1
```

Figure 222. Creating a resource control table or renaming Q.RESOURCE_TABLE

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 223 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 223. Redefining the Q.RESOURCE_VIEW

Table 79. 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.
FLOATVAL	FLOAT		Yes	Reflects resource limit for resource options that have floating point values. FLOATVAL is null for the IBM-supplied governor.

Table 79. Structure of the Q.RESOURCE_TABLE table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
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 599 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 615 explains this step.
3. Establish addressability to the QMF cancellation service to cancel activities. "Canceling user activity" on page 630 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 631 explains this step.

See the following sections for OS/390, VM, and VSE for additional steps for those particular platforms.

Modifying the governor exit on OS/390

For TSO, and native OS/390 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 OS/390" on page 639 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 80 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 xiv) that matches the NLF you're using. In the component names, a 1 represents TSO and native OS/390 batch.

Table 80. IBM-supplied governor components

Member Name	Library	Function
	TSO, ISPF, and native OS/390	
DSQUnGV1	QMF720.SDSQLOAD	Load module for TSO, and native OS/390 batch
DSQUnGV1	QMF720.SDSQUSRn	Source code for governor exit routine for TSO, and native OS/390 batch
DXEUnGV1	QMF720.SDSQUSRn	Contains text and related definitions for the governor prompts and cancellation messages in TSO, and native OS/390 batch
	CICS on OS/390	
DSQUnGV3	QMF720.SDSQLOAD	Load module for CICS
DSQUnGV3	QMF720.SDSQUSRn	Source code for governor exit routine for CICS
DXEUnGV3	QMF720.SDSQUSRn	Contains text and related definitions for the governor cancellation message in CICS
DXEUnGM	QMF720.SDSQUSRn	Contains BMS map for the governor prompts in CICS.
DXEGOVA	QMF720.SDSQUSRn	DSECT for the DXEGOVA control block
DXEXCBA	QMF720.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 OS/390 batch and the module DSQUDGV1 for German in TSO, and native OS/390 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 OS/390 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 QMF720.SDSQLOAD is assumed to be in a library concatenated to the user's STEPLIB data set.

Figure 224 shows the program structure of a governor exit routine.

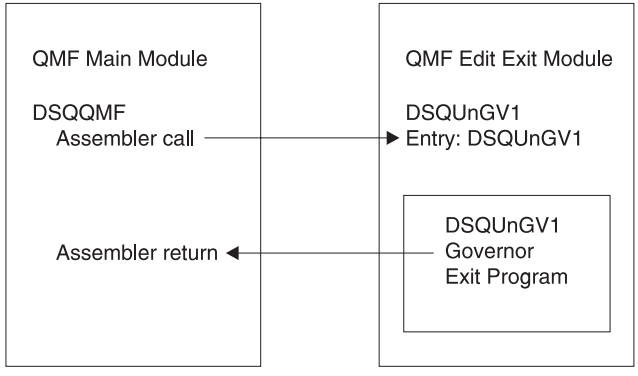


Figure 224. TSO or native OS/390 processing that interacts QMF with the governor exit

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 OS/390" on page 639.

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 226 on page 596 shows the program structure of a governor exit routine.

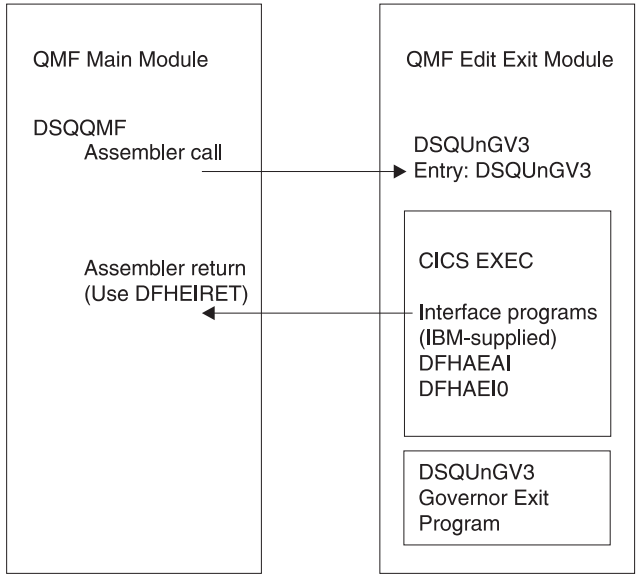


Figure 225. CICS processing that interacts QMF with the governor exit

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 OS/390" on page 639.

Modifying the governor exit on VM

On CMS, assemble and generate your governor exit routine, whether you modified the IBM-supplied governor exit routine or wrote your own.

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 81 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 xiv) that matches the NLF you're using. In the component names, a 2 represents CMS.

Table 81. IBM-supplied governor components

Member Name	Library	Function
	CMS	
DSQUnGV2	PRODUCTION DISK	Text file and member of load library
DSQUnGV2	PRODUCTION DISK	Source code for governor exit routine
DXEGOVA	DSQUSERE MACLIB	DSECT for the DXEGOVA control block.
DXEXCBA	DSQUSERE MACLIB	DSECT for the DXEXCBA con6trol block.
DSQUnGV2	DSQUSERE MACLIB	Contains text and related definitions for the governor exit routine prompts and cancellation message

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 DSQUEGV2 for English and the module DSQUDGV2 for German in.

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 CMS interacts with the governor exit routine

At the start of a user's session, QMF loads 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.

After loading the governor, QMF calls it once during session initialization. On this call, the governor should initialize itself for the user's QMF session. Toward this end, QMF passes to the governor the rows in the resource control table for the user's resource group. Resource groups and control tables were described in "How the governor knows what the resource limits are" on page 576.

Within QMF are exits, each marking the beginning or end of some activity. When control reaches one of these exits, QMF calls the governor. The first such exit is the one, just described, for the governor's initialization. The last is part of session termination. On this last call, the governor can do whatever is needed for its own termination. It might, for instance, release storage it no longer needs.

Between the first and last calls, QMF can call the governor many times from many different exits. Some of these calls, for example, precede the execution of a QMF command. The types of calls are described in detail in "How and when QMF calls the governor exit routine" on page 599.

Figure 226 shows the processing that interacts QMF with the governor exit routine:

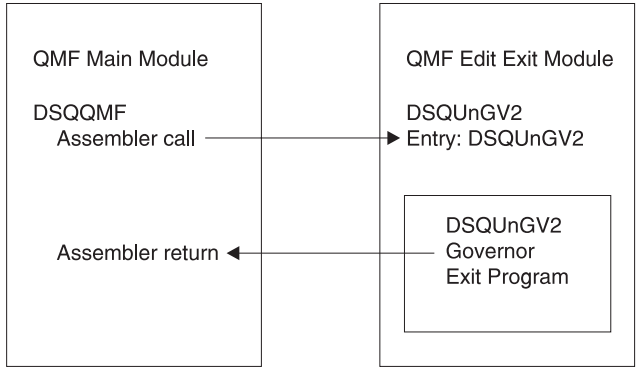


Figure 226. CMS processing that interacts QMF with the governor exit routine

Modifying the governor exit on VSE

Translate, assemble, and link-edit your governor exit routine, whether you modified the IBM-supplied governor exit routine or wrote your own.

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 82 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 xiv) that matches the NLF you're using. In the component names, a 3 represents CICS.

Member Name	Library	Function

Table 82. IBM-supplied governor components

VSE		
DSQUnGV3.PHASE	PRD2.PROD	Executable phase installed during QMF installation.
DSQUnGV3.Z	PRD2.PROD	Source code for governor exit routine.
DXEGOVA.A	PRD2.PROD	DSECT for the DXEGOVA control block.
DXEXCBA.A	PRD2.PROD	DSECT for the DXEXCBA control block.
DXEUnGV3.A	PRD2.PROD	Contains text and related definitions for the governor exit routine cancellation message in CICS.
DXEUnGM.Z	PRD2.PROD	Contains CICS basic mapping support (BMS) information, which describes how the governor prompts appear on the screen.
DSQ3nGLK.Z	PRD2.PROD	Job that translates, assembles, and link-edits the IBM-supplied governor exit routine and the BMS map.

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 DSQUEGV3 for English and the module DSQUDGV3 for German.

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 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.

The CICS control block interface to the governor exit consists of the following parts:

- Interface control blocks DXEXCBA.A and DXEGOVA.A, 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 227 shows the program structure of a governor exit routine.

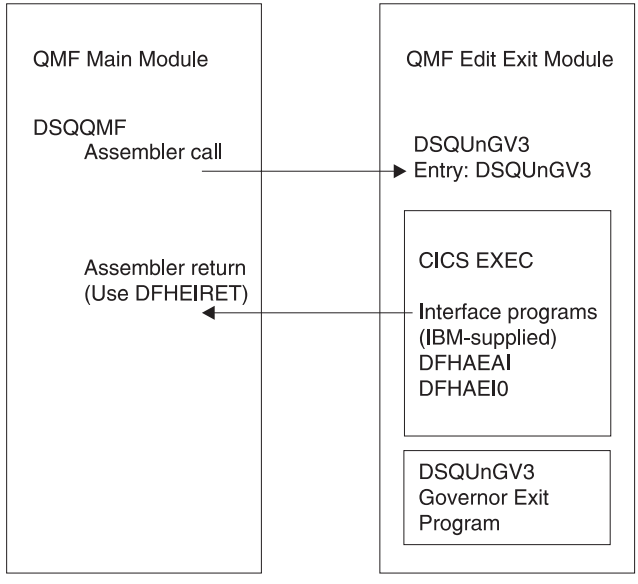


Figure 227. CICS processing that interacts QMF with the governor exit

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 OS/390" on page 639.

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.

OS/390

Follow these instructions for OS/390.

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 687 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 228 on page 602.

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 228 on page 602shows 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.

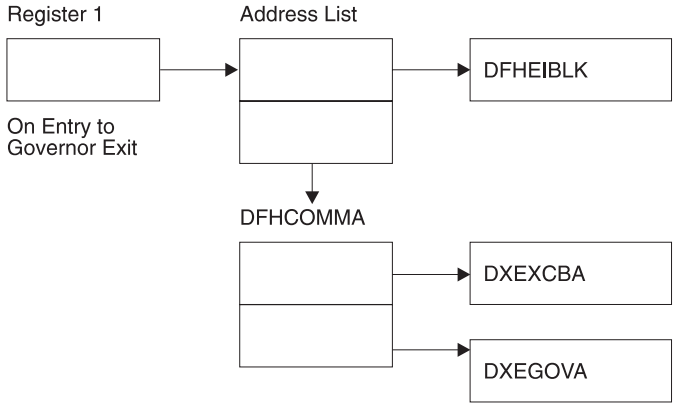


Figure 228. Contents of Register 1 on a call to the governor exit routine

- 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 229. Sample code at the start of a governor (for CICS)

The code in Figure 229 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 230 on page 604.

```
.
.
.
*
         XR    R15,R15           ZERO RETURN CODE
         DFHEIRET RCREG=15
*
```

Figure 230. 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.

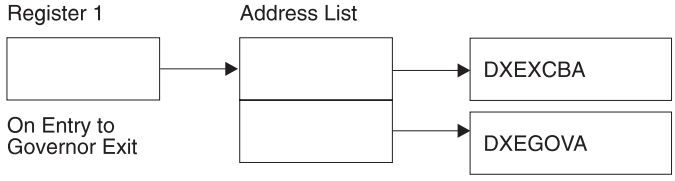


Figure 231. Contents of Register 1 on a call to the governor exit routine

- 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 232 on page 605. The code sample is from the IBM-supplied governor for TSO or native OS/390.

```
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 232. Sample code at the start of a governor (for TSO, ISPF, or native OS/390)

The code in Figure 232 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 233. (If you need more information about the activity that occurs at each function call, see "Points at which QMF calls the governor" on page 611.) 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 233. 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 234 on page 607 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 234. Identifying the type of function call and branching to the appropriate address

VM

Follow these instructions for VM.

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 687 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 the GDDM interactive chart utility panels for QMF charting functions
- Running EDIT PROC or 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.

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*GV2.

Branching to the entry point: QMF calls the governor exit routine by branching to the address of the entry point DSQUnGV2. 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.

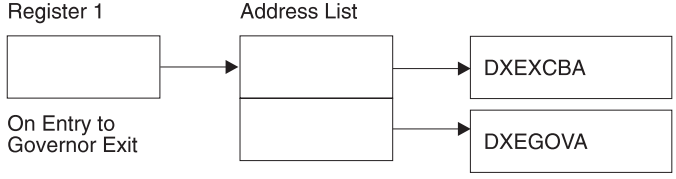


Figure 235. Contents of Register 1 on a call to the governor exit routine

- 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 DSQUnGV2.

Establishing addressability for function calls

Because QMF always branches to an entry point named DSQU*n*GV2 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 below.

```
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 236. 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 237 on page 611 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 237. Identifying the type of function call and branching to the appropriate address

Because the governor program runs on the same level as the main QMF program, ensure you preserve the QMF environment at every function call. Use the standard assembler RETURN statement to return control to QMF after every call.

VSE

Follow these instructions for VSE

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 687 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
  - Displaying the GDDM interactive chart utility panels for QMF charting functions

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.

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*GV3.

Entry to the governor exit routine follows the standard CICS linkage conventions:

- Register 1 contains a CICS parameter list suitable for processing by CICS-supplied macros DFHEIENT and DFHEIRET. Figure 238 on page 614 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.

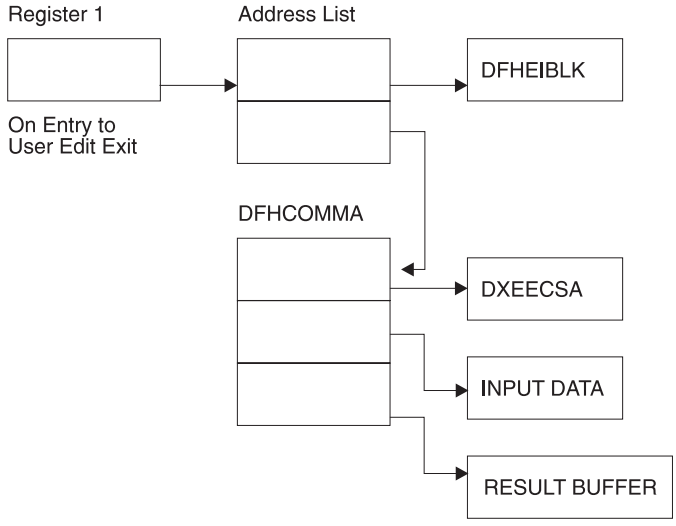


Figure 238. Contents of Register 1 on a call to the governor exit routine on VSE

- Register 13 contains the address of a standard CICS working storage area as described by CICS DSECT (DFHEISTG).
- Register 14 contains the return address.

Establishing addressability for function calls

Because QMF always branches to an entry point named DSQU*n*GV3 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 below. (If you need more information about the activity that occurs at each function call, see “Points at which QMF calls the governor” on page 611.) 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 239. 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 240 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 240. 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 sections, which describe how to set up resource control information in a format the governor can use:

- "How a governor exit routine controls resources" on page 585
- "Defining your own resource limits" on page 588

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. See "How the governor knows what the resource limits are" on page 585 for more information on how this view is used.

Table 83 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 83. 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.

Table 83. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data Type	Purpose
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.
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).

Table 83. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data Type	Purpose
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 7 RELEASE 2 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 241. 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 241. 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 242 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 242. 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 84 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 84. 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.

Table 84. 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 cancelation of the current command. The field is set to 1 if cancelation is requested. Zero indicates that no cancelation was requested. The value changes at the point at which the cancelation 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 1 for DB2 for VM or VSE and to 2 for DB2 for OS/390.
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 cancelation 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 the *Developing QMF Applications* manual.
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 84. 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 xiv.) Value does not change during a session.
XCBPANEL (ISPF only)	CL8	Contains the panel ID for the message Help panel for a cancelation 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. The query cost estimate is not available from DB2 Parallel Edition V1.2 or DataJoiner v1.2.1. In these environments the value is set to 1.
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.

Table 84. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
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 on OS/390 DB2 system tables or QMF control tables. DISPLAY TABLE EDIT TABLE ERASE TABLE EXPORT TABLE IMPORT TABLE PRINT TABLE RUN QUERY SAVE DATA
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 QMF Version 7.2, 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 OS/390 batch. Possible values are: 1 CMS (VM/SP) 3 TSO, or native OS/390 batch (MVS/XA or MVS/ESA) 4 CMS (VM/XA or VM/ESA) 5 CICS (VSE/ESA, MVS/ESA, or MVS/XA) .

Table 84. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
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 692.
XCBUSER	CL8	Contains the user's TSO logon ID (for TSO), the user parameter on the job statement (for native OS/390 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 7 RELEASE 2 LEVEL 0                           * 00010000
*                                                                      * 00011000
*        INNER CONTROL BLOCKS: NONE                                    * 00012000
*                                                                      * 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'          -------- MVS/ESA or XA (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 243. 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                   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 243. 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
```

Figure 243. 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 244 on page 630 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 244. 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 cancelation service to cancel an activity, branch to the address that appears in the DXEGOVA control block field named GOVCADDR. Figure 245 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 245. 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, native OS/390, or CMS, 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 OS/390)

or real time (CMS), and the number of rows fetched for the current DATA object. Your VM system should be running with the value of CP TIMER set to REAL. 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 OS/390 the source code is the member DSQU*n*GV1 of the library QMF720.SDSQUSRE. On CMS the source code is the file DSQU*n*GV2 on the production disk.

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.

OS/390

Your governor exit routine can cancel asynchronous commands when a timer is active in TSO or native OS/390.

To cancel QMF commands using asynchronous processing in TSO or native OS/390, 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.

VM

Your governor exit routine can cancel asynchronous commands when a timer is active in CMS.

To cancel QMF commands using asynchronous processing in CMS, 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 real time and the number of rows fetched for the current DATA object. Your VM system should be running with the value of CP TIMER set to REAL. 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 OS/390, VM, and VSE.

OS/390

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 692.

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.

VM

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 definition.

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 692.

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.

VSE

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.

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 692.

The IBM-supplied governor does not log messages for termination function calls.

Assembling and generating your governor exit routine in CMS

Whether you are modifying the IBM-supplied governor exit routine or writing a routine of your own, you need to create a CMS module.

If you are migrating from an earlier QMF release: Starting QMF from ISPF with PGM or DCSS form no longer has an effect on how to create the governor module.

Assembling your governor exit

The IBM-supplied governor is written for HLASM. To use the IBM-supplied governor, IBM supplies governor control blocks (DXEGOVA and DXEXCBA) in DSQUSERE MACLIB, which is located on the QMF production disk.

If you assemble the IBM-supplied governor, you need to issue a global maclib command for the following libraries:

1. DSQUSERE
2. OSMACRO
3. TSOMAC

For example, use the following statements to assemble the QMF-supplied governor:

```
Address CMS "PRODUCT HLASM"
Address CMS "PRODUCT QMF"
Address CMS "GLOBAL MACLIB DSQUSERE DMSSP CMSLIB OSMACRO TSOMAC "
Address CMS "HLASM DSQUEGV2
```

Building a module file or creating a load library member

After you assemble your governor, a TEXT file is created. You then need to build a relocatable module file named DSQUEGV2 or create a member of a CMS LOADLIB.

Important: If you are using your own governor, the DSQUEGV2 file can run in 31-bit addressing mode. If you are using the IBM-supplied governor, DSQUEGV2 must run in 24-bit mode. For example, use the following REXX statements to build a module file for the IBM-supplied governor:

```
Address CMS "LOAD DSQUEGV2 (RLDSAVE AMODE 24 RMODE 24"
Address CMS "GENMOD DSQUEGV2"
```

If you choose to create a member of a CMS LOADLIB:

1. Create a SYSLIN file that contains the following statements:

   ```
   INCLUDE DSQUEGV2
   ENTRY DSQUEGV2
   ```

2. Allocate the SYSLIN and INCLUDE files using the following CMS commands:

   ```
   FILEDEF SYSLIN DISK SYSLIN CONTROL A
   FILEDEF DSQUEGV2 DISK DSQUEGV2 TEXT A
   ```

3. Create the module as a member of a new or existing CMS load library using the following CMS command:

   ```
   LKED DSQUEGV2 (NCAL LET REUS NAME DSQUEGV2 LIBE USERLIB)
   ```

Assembling and link-editing your governor exit routine in TSO, ISPF, and native OS/390 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 QMF720.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 QMF720.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 QMF720.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 QMF720.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 OS/390 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 and generating your governor exit routine in CMS

Whether you are modifying the IBM-supplied governor exit routine or writing a routine of your own, you need to create a CMS module.

If you are migrating from an earlier QMF release: Starting QMF from ISPF with PGM or DCSS form no longer has an effect on how to create the governor module.

Assembling your governor exit

The IBM-supplied governor is written for HLASM. To use the IBM-supplied governor, IBM supplies governor control blocks (DXEGOVA and DXEXCBA) in DSQUSERE MACLIB, which is located on the QMF production disk.

If you assemble the IBM-supplied governor, you need to issue a global maclib command for the following libraries:

1. DSQUSERE
2. OSMACRO
3. TSOMAC

For example, use the following statements to assemble the QMF-supplied governor:

```
Address CMS "PRODUCT HLASM"
Address CMS "PRODUCT QMF"
Address CMS "GLOBAL MACLIB DSQUSERE DMSSP CMSLIB OSMACRO TSOMAC "
Address CMS "HLASM DSQUEGV2
```

Building a module file or creating a load library member

After you assemble your governor, a TEXT file is created. You then need to build a relocatable module file named DSQUEGV2 or create a member of a CMS LOADLIB.

Important: If you are using your own governor, the DSQUEGV2 file can run in 31-bit addressing mode. If you are using the IBM-supplied governor, DSQUEGV2 must run in 24-bit mode. For example, use the following REXX statements to build a module file for the IBM-supplied governor:

```
Address CMS "LOAD DSQUEGV2 (RLDSAVE AMODE 24 RMODE 24"
Address CMS "GENMOD DSQUEGV2"
```

If you choose to create a member of a CMS LOADLIB:

1. Create a SYSLIN file that contains the following statements:

   ```
   INCLUDE DSQUEGV2
   ENTRY DSQUEGV2
   ```

2. Allocate the SYSLIN and INCLUDE files using the following CMS commands:

```
FILEDEF SYSLIN DISK SYSLIN CONTROL A
FILEDEF DSQUEGV2 DISK DSQUEGV2 TEXT A
```

3. Create the module as a member of a new or existing CMS load library using the following CMS command:

```
LKED DSQUEGV2 (NCAL LET REUS NAME DSQUEGV2 LIBE USERLIB)
```

Assembling, translating, and link-editing your governor exit routine in CICS on OS/390

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 QMF720.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 QMF720.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 QMF720.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 QMF720.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)
```

Assembling, translating, and link-editing your governor exit routine in CICS on VSE

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

Before you assemble your governor exit routine, establish a VSE library exit to handle macro processing of E-decks. The *VSE Guide to System Functions* provides a description on how to establish this exit.

Use the HLASM compiler option in the following example to assemble to routine. The LIBEXIT parameter includes CICS macro definitions created by the CICS translation process.

```
'LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXIT(ORDER=EA)))'
```

In the source library search specification, specify the QMF governor exit interface control blocks DXEXCBA.A and DXEGOVA.A located in the QMF library.

Link editing your governor exit routine

Create a new QMF governor exit routine named DSQU*n*V3 by including the EXEC CICS interface control modules DFHEAI and DFHEAI0, and you governor exit program DSQU*n*V3. The EXEC CICS module DFHEAI must be the first module in your governor exit routine, and the entry point must be

the QMF module DSQU*n*V3. DSQU*n*V3 must be executable in 31-bit addressing mode. Replace the *n* symbol with an NLID that corresponds to the national language that you are using.

Example JCL statements

Figure 246 on page 642 shows the JCL used to install, translate, assemble, and link-edit the IBM-supplied governor exit routine. This JCL is supplied in the QMF sublibrary, under the name DSQ3GV3.Z. For more information on installing your own program into CICS, see the *CICS System Definition Guide*.

```
...* $$ JOB JNM=DSQ3GV3,DISP=D,CLASS=0
// JOB DSQ3GV3 Sample Job to Install Cutomer Written QMF Governor
* ---------------------------------------------------------------
* Instal QMF Governor Exit (HLASM)
* ---------------------------------------------------------------
// SETPARM VOLID=volid  *--update volid for syspch
// SETPARM START=rtrk   *--update start track/block (syspch)
// SETPARM SIZE=ntrks   *--update number of tracks/blocks (syspch)
* ---------------------------------------------------------------
* Library search chain must contain QMF, CICS, and HLASM sublibrary
* ---------------------------------------------------------------
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE)
// LIBDEF PHASE,CATALOG=PRD2.PROD
* ---------------------------------------------------------------
* Step 1: Translate Governor exit program
* ---------------------------------------------------------------
// DLBL   IJSYSPH,'ASM.TRANSLATION'.0
// EXTENT SYSPCH,,1,0,&START.,&SIZE.
ASSIGN SYSPCH,DISK,VOL=&VOLID.,SHR
// EXEC   DFHEAP1$
         :
         :
         Your governor program
         :
         :
/*
* ---------------------------------------------------------------
* Step 2: Assemble Governor exit program
* ---------------------------------------------------------------
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'ASM.TRANSLATION',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=&VOLID.,SHR
// OPTI0N CATAL,DECK,SYM,ERRS
   PHASE DSQUEGV3,*,SVA
         INCLUDE DFHEAI
         INCLUDE DFHEAI0
// EXEC ASMA90,SIZE=(ASMA90,50K),                                C
              PARM='LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXITC
              (ORDER=EA)))'
CLOSE SYSIPT,SYSRDR
/*
* ---------------------------------------------------------------
* Step 3: Link-edit Goernor exit program
* ---------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D       IF 1A93D, CLOSE SYSPCH,00D
/&
...* $$ EOJ
```

Figure 246. Example JCL for translating, assembling, and link-editing a governor exit

Using the DB2 governor on OS/390

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 UDB for OS390 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.

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 for

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:

- An authorization ID
- The name of a DB2 application plan
- A value for the maximum processor time
- The LU name of the site where the request originated (for DB2 2.2 and above)

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 and the name of the QMF application plan. The row for the other users contains the QMF plan name and 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 the *DB2 UDB for OS390 Administration Guide* .

Chapter 31. 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 the *Using QMF* manual.

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 batch a batch program on OS/390

This chapter section how you can use the QMF batch mode in TSO, ISPF, native OS/390, or CICS. For ISPF on OS/390, the QMF batch facility executes QMF in the TSO terminal monitor program (TMP).

TSO

The order of information for OS/390 is: TSO, ISPF, native OS/390, 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 651.

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 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 OS/390 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 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 247. 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 22, "Customizing Your Start Procedure", on page 259.
- 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 21, "Starting QMF", on page 237. 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 27, "Customizing QMF Commands", on page 457 for the procedure on how to assign synonyms.

Assigning authority to use the application on OS/390

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 374.)

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 248 on page 655

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 248. 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 85 describes the required fields.

Table 85. 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 85. 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 86 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 86. 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 86. 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 86. 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 OS/390 JCL and must conform to the OS/390 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, OS/390 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 86. 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 QMF720.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 QMF720.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 QMF720.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 QMF720.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 249 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 249. Modifying the DSQABB1S SKELETON

Make the five modifications to DSQABB12 CLIST as commented in Figure 250 on page 662.

```
/*************************************************************/   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 250. 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 250. Modifying DSQABB12 CLIST (Part 2 of 2)

Running QMF batch in native OS/390

In addition to running QMF batch in TSO and ISPF, you can run QMF as a native OS/390 batch job. You can use the JCL shown in Figure 251 on page 664 to run QMF as a batch job in native OS/390.

```
/************************************************************/      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=QMF720.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=QMF720.DSQMAPE,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 251. JCL to run QMF as a native OS/390 batch job

When you run QMF in native OS/390, remember these facts:

- TSO is not available.
- QMF functions that require TSO or ISPF will not work when you run QMF in native OS/390.
- 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 252 on page 665.

The REXX program listed in Figure 252 on page 665 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.QMF720.SDSQLOAD,DISP=SHR
//ADMGGMAP  DD  DSN=QMFDEV.QF720.DSQMAPE,DISP=SHR
//SYSEXEC   DD  DSN=ROBIN.QMF720.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 252. REXX program to start and run QMF in batch mode

Running QMF as a non-interactive transaction on CICS

This CICS procedure works on both OS/390 and VSE. In 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 253 on page 666 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 253. 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.

Running QMF as a batch program on CMS

Authority to operate in batch mode

Anyone can send a job to the CMS batch machine. Either the batch machine can use the minimum authority for running the job granted to that batch machine's ID, or it can use the authorization granted to some user through the CONNECT command. The CONNECT command must be used to enable a user to access data in batch mode with the same authorization that user has when working with QMF interactively.

If you or your users create a procedure to run in batch, and save that procedure with SHARE=YES, anyone can display it. If the procedure also has your CONNECT ID in it, then anyone who can display that procedure can see your CONNECT ID and its associated password.

A batch machine's authority depends on your installation setup. It is possible for users to run a job and save queries, procedures, and forms under the batch machine's ID. If users are allowed to save things under the batch machine's ID instead of their own IDs, you or the database administrator must clean up the database periodically and purge what is owned by the batch machine. If items are saved in this way, let users know that what they save under the ID of the batch machine might be purged from the database on a periodic basis.

To provide a user with CONNECT authority, you must grant them access using the following query:

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

You need to ensure that your own or your user's IDs are not made public.

You can use one of the following procedures:

- The job you send to the CMS batch machine can call an exec that creates a procedure on the CMS batch machine's A disk containing the CONNECT ID, the CONNECT ID password, and a command to run the user's procedure. This intermediary procedure, created by the exec, then connects to the database and runs the user's procedure already saved in the database. The procedure named on the DSQSRUN parameter of the ISPSTART command imports the intermediary procedure that connects to the database and runs the user's procedure.
- You can send the data in-stream (with the CMS batch job) and use the CMS `MOVEFILE` command. This process creates a procedure containing the CONNECT ID, the CONNECT ID password, and a command to run the user's procedure.

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.

Sending a job to the CMS batch machine

Users can execute procedures in batch mode by sending jobs to the CMS batch machine. They can then continue their sessions without waiting for those procedures to be run. For example, a user can send the following exec and continue to work:

```
/* Sends batch job file to batch machine                          */
/*      Syntax:  BATCHJOB fn ft <fm  <batmach>>                   */
/*          where batmach is the name of the batch machine        */
/*                          (default is CMSBATCH)                 */
/*                                                                */
Parse upper arg fn ft fm batmach
If batmach = '' then batmach = 'CMSBATCH'
'PUSH PUN'
'CP SPOOL PUN NOHOLD NOCONT TO' batmach
'PUNCH ' fn ft fm
'POP PUN'
```

Figure 254. Sample EXEC to send a job to the batch machine

You also need to tell the CMS batch machine to execute the procedure in batch mode.

Figure 255 on page 669 is a sample job to start QMF in batch mode. Lowercase words in it are parts of commands filled in by you.

```
/*
/JOB userid acctnum jobname
/SET TIME 10 PUNCH 3000 PRINT 3000

*---Spool PRINTER, PUNCH and CONSOLE to userid
CP SPOOL 00E CONT DIST userid TO userid NOHOLD
CP SPOOL 009 NOHOLD TO userid
CP SPOOL 00D NOHOLD TO userid
CP SPOOL 009 START

*--- Link to userid's disk
CP LINK userid 191 192 RR readpass
ACCESS 192 B/A

*-------------------------------------------------------------*
* Tailor the QMF invocation EXEC DSQ2EINV which
* first links to GDDM, ISPF, DB2 for VM, QMF, and then
* creates the FILEDEFs to run the job.
* The result from this tailoring can be invoked
* as an EXEC or can be coded in line with this
* sample job.
*-------------------------------------------------------------*

* QMF invocation follows:
* Run with code in BATCHMODE, and pass the name
* of an invocation QMF PROC to run.
* EXEC ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE)
*     PARM(dcssname(DSQSMODE=B,DSQSRUN=userid.myproc))
* Other forms of QMF invocation which can be used are as follows:
* DSQQMFE dcssname(DSQSMODE=B,DSQSRUN=userid.myproc))
*  when QMF is started independent of ISPF
* EXEC ISPSTART PGM(appl_name)
*  where "appl_name" is the name of the application program.
* EXEC ISPSTART DCSS(dcssname) NEWAPPL(DSQE)
*     PARM(DSQSMODE=B,DSQSRUN=userid.myproc))
* See
Chapter 22, "Customizing Your Start Procedure", on page 259 for more
* information on these forms of QMF invocation.
*-------------------------------------------------------------*
*  MAKE THE NLF RESOURCES NEEDED FOR THE RUN AVAILABLE
*-------------------------------------------------------------*
EXEC QMFBATCH DEUTSCH LMN.PROCA
*---Close the PRINTER-----------------------------------------*
CP SPOOL 00E NOCONT
CP CLOSE 00E
* You can also run an application in batch without ISPF.
* You would use the following command:
* EXEC MYAPEXEC
/*
```

Figure 255. Sample job to send to CMS batch machine

The job first spools the printer, punch, and console, and then accesses the user's A-disk as an extension of the batch machine's A-disk. Having done this, it invokes an exec to allocate the necessary resources and start QMF.

Sample job notes:

1. Using parameter 'DSQSMODE=B' to indicate batch mode means you must include the 'DSQSRUN' parameter (as in the preceding example) to name the procedure you run.
2. The CMS batch machine must be authorized to both start QMF and to connect to DB2 for VM.
3. The DCSS name following PARM on the ISPSTART command must be included if it is anything other than the default: QMF720E. It is required if you use the DCSS(dcssname) form of the end.
4. The language exec included is started by the following statement:

   ```
   EXEC QMFBATCH language proc
   ```

   where language is the language for the session and proc is the name of the QMF procedure to be run.

Running batch jobs on your CMS machine

You can start QMF to run a job in batch mode (DSQSMODE=B) without sending the job to the CMS batch machine. Invoking QMF for such a job means that QMF comes up and runs the procedure you specify with the DSQSRUN parameter.

For more information on passing parameters to QMF, see Chapter 22, "Customizing Your Start Procedure", on page 259. Before doing this however, check the ISPF profile and ensure a value is entered for CONSOLE PROCESS OPTION. you for a CONSOLE PROCESS OPTION value.

Debugging a procedure

You can use both the trace codes and the tool to diagnose problems with a batch mode procedure. L2 tracing is the default for procedures run in batch mode. A SET command in your procedure will change the trace level. For example, to specify L1 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 DSQDEBUG. If the procedure terminates prematurely, an error messages logged to the DSDQDEBUG data set. Use the HELP command to reconstruct the corresponding message HELP panel.

MACLIBs required on VM

The following MACLIBs must exist for use by this application. Appropriate FILEDEFs for these MACLIBs should be named on the exec used to invoke ISPF.

Note: In addition, several other files are supplied with QMF. If you want to examine them, they can be printed from the distribution disk:

- DSQMLIBE

  This is the application message library. It should be concatenated with ISPMLIB. For this application, the members of this MACLIB are DSQBE00, DSQBE01, and DSQBE02.

- DSQPLIBE

  This is the library containing the application's panels. It should be concatenated with ISPPLIB DXYEABVP, DXYEABV1, DXYEABV2, and DXYEABV3 are the members of this library for this application.

- DSQSLIBE

  This is the library containing the application skeleton library. It should be concatenated with ISPPLIB. DSQABB2P, DSQABB2J, and DSQABB2S are the members of this library for this application.

Using 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
- 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.

To use the batch application, enter:

```
BATCH
```

which results in the display of the prompt panel in Figure 256 on page 672

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
FORM  NAME     ===>                 Form to be used with query
Current FORM   ===>  NO             Use form in temporary storage?
BATCH NAME     ===>                 Name of QMF batch execution proc
PROC arguments ===> ARGS
CONNECT PASSWORD    ===>            Database password
DISK PASSWORD       ===>            User 'A' disk read pasword
LOGGING             ===>  YES       Log messages and commands?
BATCH MACHINE       ===>            CMS ID of batch machine
SAVE DATA           ===>            Name of data to be saved
REVIEW OUTPUT       ===>  YES       Send report to your reader?
DISTRIBUTION        Userids and notes to send report
      USERID        ===>                  NODE  ===>
                    ===>                        ===>
PRINT OUTPUT:       Printer ID and node for printed output.
         ID         ===>                NODE     ===>
PF1=Help PF3=End  Enter=Process batch request
```

Figure 256. 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 87 describes the required fields.

Table 87. 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 87. 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.
PROC arguments	Through this field, you can pass arguments to the REXX procedure specified in the OBJECT NAME field.
CONNECT PASSWORD	Users are required to enter the DB2 for VM password. Assign this to a user in the SYSTEM.SYSUSERAUTH table. This password is used in a CONNECT command in the batch machine. The user is then operating with the authority granted to the DB2 for VM user ID. The batch procedure is run with this authority.
DISK PASSWORD	Users are required to enter their 191 A-disk read password. (If the user has no read password, 'ALL' must be entered instead.) This is used in the batch job sent to the CMS batch machine. The batch machine then links to the user's 191 disk.
BATCH MACHINE	Users are required to enter the CMS user ID of a batch machine on which the job is to be run. The job is punched to this machine. This value is saved across sessions for users. The batch machine must exist on the same processor as that of the user.

Optional entry fields

Table 88 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 88. 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.

Table 88. 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.
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 doesn't 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.
REVIEW OUTPUT	If the user wants to view the report from running the batch query or procedure, YES (the default value) should be specified as the value for this field. The report is sent to the user's reader using SENDFILE. If the query or procedure to be sent to batch does not generate a report, such as an INSERT or UPDATE query, this field should be set to NO.

Table 88. 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. The report is sent using SENDFILE, which makes use of the NAMES file. Because of this fact, the NODE need only be supplied if the recipient of the report is on a different system and there is no entry for that user in the NAMES file. The USERID can also be a list defined in the NAMES file. If the query or procedure to be sent to batch does not generate a report, such as an INSERT or UPDATE query, no values should be supplied for these fields.
PRINT OUTPUT	If the user wants the resulting report to be sent to a printer, the printer ID and the NODE should be entered here. If the printer ID is SYSTEM, the output is sent to the system printer. The appropriate CP SPOOL and TAG commands are executed before the report is printed.

Modifying the batch application

When you make modifications to the application, be sure to save a copy of the original file. Modify a copy of an original file that has been renamed. Keep a backup copy of any original file and its modified version. This way, a new copy sent by IBM will not replace it.

Three modifiable model files are shipped with the product. They provide input to ISPF file tailoring, which in turn produces three files needed to run this application. One of these model files, DSQABB2J COPY, is the skeleton file behind the actual job sent to the CMS batch machine. In DSQABB2J COPY, you can modify the following:

- The account number
- The print and punch output limits
- The maximum processor time allowed for a job
- The name of the discontiguous shared segment (DCSS) on the ISPSTART command
- The SQLINIT statement to specify another database if queries are run in some database other than SQLDBA
- The links to the product disks

The two other model files do the following:

- DSQABB2P COPY creates the user's batch procedure.
- DSQABB2S COPY saves a user's query, form, and procedure and punches a job to the CMS batch machine. It also erases any work files that were created.

Chapter 32. 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 689 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 the *QMF Messages and Codes* manual.
- 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.

OS/390 concerns

If any DSN (DB2) and ADM (GDDM) error messages appear, look them up in the *QMF Messages and Codes* manual.

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.

VM concerns

Follow these instructions in addition to the general ″Handling initialization errors″ instructions.

- 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.

  If users try to start QMF through ISPF, and QMF fails to start, the following message appears:

  ```
  INITIAL PGM RC ¬ = 0 | 4 - THE INITIAΛΛΨ INςOKEΔ MOΔΥΛE
   ENΔEΔ ΩITH A PETΥPN XOΔE = 16 .
  ```

- 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 CMS command SET EMSG ON and restart QMF.

VSE concerns

Check that the DB2 database is initialized and working properly.

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 27, “Customizing QMF Commands”, on page 457
  - The structure for function key tables is shown in Chapter 28, “Customizing QMF Function Keys”, on page 479
- All rows of the table contain valid data. If this might be the problem, see:
  - “Entering command synonym definitions into the table” on page 465 for information on valid command synonym definitions

  - "Entering your function key definitions into the table" on page 484 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.

OS/390 concerns

More information about the error is logged in the user's trace data. In TSO and native OS/390, 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.

VM concerns

More information about the error is logged in the user's trace data. In CMS, the trace data is stored in DSQDEBUG.

VSE concerns

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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual.

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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual.

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 OS/390

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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual.

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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual.

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 VM

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.

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. .

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.

Handling GDDM errors on VSE

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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual.

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 the *GDDM System Customization and Administration* manual or the *GDDM Installation and System Management for OS/390* manual.

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 OS/390

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.
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 QMF errors during printing on VM

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.

What happens	What it means	What to do
After using the CONNECT command, your PRINT commands result in the message described, or your printed output goes to a different printer.	The CONNECT command replaces your own profile values with those of the user you connected to.	After connecting, remember to enter: SET PROFILE (PRINTER=prtname from the command line to establish your printer name as the default.
You issue several PRINT commands but find that only the last object is printed.	Your output data set does not have a disposition of MOD, so each PRINT operation reopens the data set and overwrites the previous contents.	Change the disposition of your output data set to MOD. You cannot use the disposition MOD with a member of a regioned data set.
You print a QMF object and see unexpected control characters in the printed output or data set.	The device token or PROCOPT you are using does not match the device on which you are actually printing.	Supply the correct device token, or reduce control characters to a minimum by one of these techniques: • 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 CMS command errors

You might encounter problems when using the QMF CMS command in the following ways:

- When using The CMS command to run an exec
- When using the CMS command if QMF has been started using ISPF
- If the CMS command is used to invoke a function and that function executes a program that issues a DB2 for VM CONNECT
- If the CMS command is used to invoke a function and that function executes a program that issues a DB2 for VM COMMIT

The following sections describe the type of problem that might occur.

Using the CMS command to run an exec

QMF uses a STAE exit to establish an abend handler. If you use the CMS command to run an exec that alters the Stae exit, you can encounter the following problems:

- If the Stae exit is removed, you are not able to record abend information should a QMF abend occur.

- If a Stae exit is added, the wrong Stae exit can get control should a QMF abend occur.

Issuing the CMS command if QMF is started using ISPF

If you invoke QMF through the PGM form of the ISPSTART command, QMF packages the CMS command and uses the ISPF "SELECT CMD" service. The command is then executed in CMS subset mode. Some CMS functions do not work while in subset mode. if a function is started using the QMF CMS command and that function changes the CMS environment or resets CMS subset mode, results can be unpredictable when returning to QMF.

Note:if QMF is invoked with the DCSS form of the ISPSTART command, you do not get CMS in subset mode; you get the full CMS with all CMS functions available.

Using the DB2 for VM CONNECT command

If the CMS command is used to invoke a function and that function in turn executes a program that issues an DB2 for VM CONNECT, the results of that function are not known to QMF. In such a case when control is returned to QMF, QMF is unknowingly executing on behalf of the user ID specified by the CONNECT done outside of QMF. In this case all table requests are performed using the connect ID outside of QMF and all QMF objects are processed using the connect ID known to QMF.

Caution your end users not to use the DB2 for VM CONNECT command through the CMS command.

Using the DB2 for VM COMMIT command

If a function is invoked through the CMS command, and that function in turn executes a program that issues an DB2 forVM COMMIT command, the results can prematurely close the cursor on a QMF report object.

This might happen if the QMF report is not complete when issuing the CMS command. To prevent this from happening, complete or reset the report object prior to executing a function through the CMS command that causes a database commit. If the report cursor is closed prematurely, and you subsequently scroll to the bottom of the report, a system error occurs.

Handling display errors on VSE

The following message indicates the the object the user is trying to print needs a printer name, which QMF cannot find in the user's profile or as a default name:

```
GDDM printer nickname is required for PRINTER
```

Users who see this message should press the Enter key to display a prompt panel on which they can enter a printer name and other print parameters. Update your user profile with a valid printer nickname so QMF does not display this message again.

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 the *QMF Reference* manual.

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 29, “Creating Your Own Edit Codes for QMF Forms”, on page 497 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.

OS/390 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 the *DB2 UDB for OS390 SQL Reference* manual.

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.

VM concerns

There are no additional concerns on VM.

VSE concerns

Follow these instructions for increasing the CICS region on VSE.

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 677, use this section to find out which QMF and CMS diagnosis aids can help you to determine the problem.

Choosing the right diagnosis aid for the symptoms

Use Table 89 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, TSO, or CMS status information, and QMF messages and help to determine the problem.

Table 89. 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	OS/390, CMS, CICS/VSE	CMS	OS/390, CICS/VSE	OS/390, CICS/VSE		CMS	CMS
Batch session	OS/390, CMS	OS/390, CMS		OS/390, CMS		OS/390, CMS	OS/390, CMS
Callable interface	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE	
Display panel	OS/390, CMS, CICS/VSE	OS/390, CMS, CISS/VSE			OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Document interface	OS/390, CMS	OS/390, CMS			OS/390	OS/390	OS/390
Error messages	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE			OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Governor exit routine	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	

Table 89. 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
Incorrect output	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE			OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Initialization	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Installation	OS/390, CMS, CICS/VSE				OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Interrupt facility	OS/390, CMS	OS/390, CMS			OS/390	OS/390	OS/390
Loop		OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Performance	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Printing	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
QMF command	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE			OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
SQL error codes	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE			OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
Termination	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE		OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE	OS/390, CMS, CICS/VSE
User edit routine	OS/390, CICS/VSE	OS/390, CICS/VSE	OS/390, CICS/VSE	OS/390, CICS/VSE		OS/390, CICS/VSE	OS/390

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 the *QMF Messages and Codes* manual, 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 90. The trace IDs are the same IDs that you use to trace QMF activity for each function.

Table 90. 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
Report formatting	F	DSQ60000 - DSQ69999
Charting	P	DSQ70000 - DSQ79999
Full-screen windows	G	DSQ80000 - DSQ89999

In addition to the message numbers in Table 90, 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 the *QMF Messages and Codes* manual.

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 the *DB2 Messages and Codes* manual 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 OS/390

Follow these instructions for using the trace facility on OS/390.

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 257. 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 258 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.
&01:  DATE
&02:  STAFF
&09:  -205
----------------------------------------------------------------------
```

Figure 258. 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 258 appears in *QMF Messages and Codes*as:

```
Column &01 is not in table &02.
```

The bottom half of Figure 258shows 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 259.

```
* 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 259. 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.

Using the trace facility on VM

Follow these instructions to use the trace facility on VM.

Allocating the trace fIle on CMS: When you are using procedures involving trace information, ensure that the trace file is allocated before you begin the QMF session. This is true if the file is allocated by the PROFILE EXEC for a user ID. The default file name is DSQDEBUG.

Check with your VM administrator if you are not sure whether the file is allocated automatically before a QMF session. If it is not, issue the following CMS statement before you start QMF for your diagnostic session.

```
FILEDEF DSQDEBUG PRINTER (LRECL 121 RECFM FA PERM)
```

If the PROFILE EXEC takes you to QMF immediately after logon and logs you off VM when you terminate the QMF session, insert the preceding FILEDEF statement into the user's PROFILE EXEC file.

Starting the trace facility:

1. Allocate a file with a file name 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. For more information on choosing trace options, see "Getting the right level of detail in your trace output" on page 694.

   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.
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.
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 260 on page 701 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.
&01:  DATE
&02:  STAFF
&09:  -205
--------------------------------------------------------------------------
```

Figure 260. 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 260 appears in *QMF Messages and Codes*as:

```
Column &01 is not in table &02.
```

The bottom half of Figure 260shows 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 might have been allocated automatically through your PROFILE EXEC. You might want to 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 actually want to display it.

To allocate (or reallocate) for printing, issue the following statement which defines DSQDEBUG as a print file:

```
FILEDEF DSQDEBUG PRINTER (LRECL 121 FA PERM)
```

The allocation contains fixed-length, 121-character records hose 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.

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.

```
FILEDEF DSQDEBUG DISK DEBUG LIST (PERM RECFM FBA LRECL 81
```

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.

Using the trace facility on VSE

Follow these instructions to use the trace facility on VSE.

Allocating storage for trace data: Choose either a CICS temporary storage or transient data queue to store trace data. If the trace data for the user's session does not exceed 32,767 rows, you can use CICS temporary storage or intrapartition transient data queues to contain it. If the trace data exceed 32,767 rows, define in the CICS DCT an extrapartition transient data queue that routes the output to a VSE file or SYSLST.

To define a transient data queue, update the CICS DCT with a 1-byte to 7-byte entry that points to the location that receives your trace data.

Figure 261 on page 704 shows the definitions for the default queue, a transient data queue named DSQD that is allocated to a SYSLST. The default location is DSQDBUG.

```
DFHDCT  TYPE=EXTRA,     QUEUE FOR QMF EXTRA PROCESSING
        DESTID=DSQD,
        RSL=PUBLIC,
        DSCNAME=DSQDBUG

DFHDCT  TYPE=SDSCI,                   DCT ENTRY FOR DEBUG OF QMF
        DSCNAME=DSQDBUG,
        RECFORM=VARUNB,
        BLKSIZE=136,
        TYPEFLE=OUTPUT,
        CTLCHR=ASA,
        DEVADDR=SYSLST,
        DEVICE=1403
```

Figure 261. Describing a SYSLST to contain trace data

Use DSQSDBQT parameter if you want to use a temporary storage queue for the trace data when you start QMF. If you want to name the queue something other than DSQD, use the DSQSDBQN parameter.

The trace data queue can be shared by all users in the same CICS partition, because QMF issues CICS ENQ and DEQ commands around single trace entries. Because tracing is an aspect of a user's profile, you can also set the level of trace detail individually for each user with the SET PROFILE command using the TRACE keyword. Records in the trace data identify individual terminal IDs for different QMF sessions on the header line.

Starting the trace facility: To start the trace facility, do one of the following tasks:

- 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.
- 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.

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 262 on page 706 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.
&01:  DATE
&02:  STAFF
&09:  -205
------------------------------------------------------------------------
```

Figure 262. 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 262 appears in *QMF Messages and Codes*as:

```
Column &01 is not in table &02.
```

The bottom half of Figure 262shows 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: Depending on the number of users and the levels of detail at which their sessions are traced, the data might be very long. Browse the data before you decide to print it.

Viewing data in a temporary storage queue: Use the CICS transaction CEBR to browse a temporary storage queue. For example, to browse a queue named MYTRACE, enter the following command from a cleared CICS screen:

```
CEBR MYTRACE
```

If the trace output is less than 32,767 rows, use temporary storage queues to hold the trace data. If the output is more than 32,767 rows, you must use a transient data queue for the trace data.

Viewing data in a transient data queue: The default queue for trace data is a transient data queue named DSQD. Trace output routed to this queue goes to the SYSLST, and can be found in the list output of your CICS job. To transfer the data from the CICS LST queue to the SYSLT so you can view it, you must stop CICS. This will allow you to browse or print the SYSLST using VSE POWER, ICCF, or another facility available to you.

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, OS/390, CMS, 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 263 shows an example.

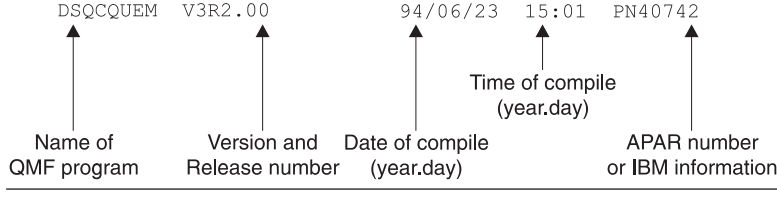


Figure 263. Example of a stamp that identifies a QMF program

Using OS/390 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 264 on page 710 shows an example of diagnostic display attributes you might use.

The definition shown in Figure 264 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 264. TYPETERM specification for CICS diagnostic display

Abend handling on VM Here

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 the problem diagnosis.

For the user edit code, DXEECS, the input area and the result area are added to the output.

Abend handling on VSE

First Paragraph

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 264 on page 710 shows an example of diagnostic display attributes you might use.

The definition shown in Figure 265 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 265. TYPETERM specification for CICS diagnostic display

Using the QMF interrupt facility

This section applies only to OS/390 and VM.

OS/390

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 266. 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 267. 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 268 on page 713

```
-- 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 268. 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 268 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.

VM

To use the QMF interrupt handler in CMS, make sure that your break key is set to PA1. To do this, enter:

```
CMS Q TERMINAL
```

Examine the BRKKEY field that is displayed. If it does not show BRKKEY PA1 , then enter:

```
CMS TERM BRKKEY PA1
```

to set the break key to PA1. Press the CLEAR key to see the QMF procedure panel.

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 269. QMF interrupt handler prompt 1

Note: If you have to use the PA1 key to create the interrupt, you might have to press the PA1 key twice before this message appears.

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 270. QMF interrupt handler prompt 2

Note: You might have to press the Enter key twice before this message appears on your screen.

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 below:

```
-- 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 271. 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 271 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.

Error handling: QMF handles interrupts through the use of a STAX exit. If you use a CMS command to execute an EXEC that alters the STAX exit, you encounter the following problems when returning to QMF:

- If the STAX exit is removed, you are not able to interrupt QMF. This means you won't be able to cancel a QMF command. You can, however, get into CP READ and issue an HX command. This action halts execution of both QMF and ISPF.
- If a STAX exit is added, you can have a problem canceling a QMF command. You can end up in the wrong interrupt handler! Here again, you can get into CP READ and issue an HX command, which halts the execution of both QMF and ISPF.

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 91.

Table 91. 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	CHAR	8	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 272 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 272. 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 719.

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 OS/390 6 begins with the component ID 566872101 and a release number (shown in Table 92) that matches the QMF national language environment in which you experienced the problem.

Table 92. 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
Swedish	65C
Swiss French	65D
Swiss German	65E
Uppercase English	651

The flowchart in Figure 273 shows how to develop your search words as you determine each characteristic of the problem.

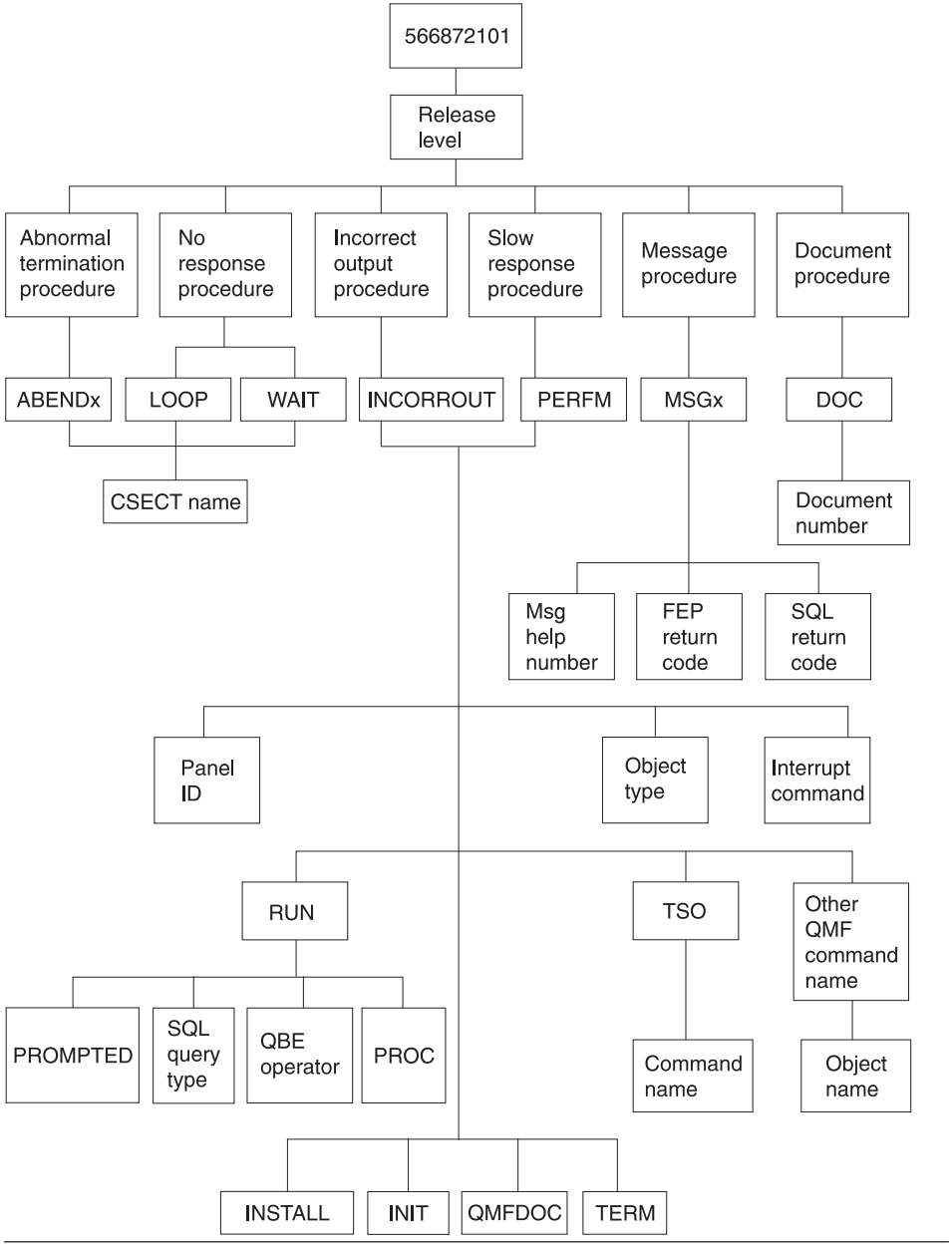


Figure 273. Chart of keyword types. Move from the top to the bottom of this chart to determine your keywords.

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 692 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 692. The IBM representative might also need documentation produced by other diagnosis aids shown in Table 89 on page 689. This documentation can help the representative recreate the problem.

Part 5. Appendixes

Appendix A. Miscellaneous

What if it did not work? (OS/390)

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 7.2 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 OS/390 interface module DSQIELI.
- In module DSQCBST CSECT DSQCMCVP

  After QMF Service has been applied, verify that an OS/390 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 "Step 24—Update CICS startup job stream" on page 73.

Look for console error messages related to the DSQPNLE data set.

AEY9 ABEND

The DB2 UDB for OS/390 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 51 for more details.

DSQ10493

This message indicates a database authorization error. Verify that the DB2 UDB for OS/390 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=QMF720,AUTH=DEPT1
```

In this example, the authorization ID is DEPT1, and the plan ID is QMF720.

DSQ36805

SQLCODE 805. This error occurs during startup. Record all the tokens returned from the SQLCODE 805, and follow the directions in the *DB2 UDB for OS390 Message and Codes* manual 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 26, "Enabling Users to Print Objects", on page 423. 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 "Step 23—Tailor the QMF profile" on page 73.

- 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 7 Release 2, but instead points to another release. In this case, make sure your ADMGGMAP ddname points to the QMF720.DSQMAPn data set. For further details on troubleshooting in general and incorrect output specifically, see Chapter 32, "Troubleshooting and Problem Diagnosis", on page 677.

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 274 on page 727.

```
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 274. 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.

QMF for CICS on VSE/ESA and OS/390 Version 7.2 product limitations

Some functions provided by QMF are dependent on underlying system services and other program products that are available in VM/CMS and OS/390/TSO, but not in CICS/VSE or CICS on OS/390. ISPF is not available in CICS. REXX is not available in QMF CICS, even though REXX is available in VSE/ESA 1.3. The following QMF functions or programs are not supported in QMF for VSE/ESA Version 7.2. 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. Other products are not available in QMF for VSE/ESA Version 7.2:

- CMS command (VM only)
- TSO command (TSO only)
- CONNECT command (when issued to connect to another database)
- Remote unit of work and distributed unit of work

  DB2 on VSE is a server, not a requester. It can be accessed by other application requesters where QMF is installed, but QMF users on VSE cannot connect to another application server with the QMF CONNECT command.
- QMF client/server components

  The function of QMF on VSE is synchronized with that of QMF on other platforms. As a result, some functions that exist in QMF Version 1 are not supported in QMF for VSE/ESA Version 7.2:
  - Command canceling
  - IMPORT ISQL queries
  - Table plot utility
  - QMF VSE Version 1 defaults module for starting QMF
  - VSE/POWER support (use methods supplied by CICS or GDDM to print your objects)
  - QMF supplied views
  - QMF sample queries from QMF VSE V1. QMF for VSE/ESA

    Version 7.2 does not supply sample objects except for tables (sample queries are not discussed in the documentation). It is not abnormal that, under some circumstances, a QMF Version 1 report looks slightly different in Version 7.2. You can make it look the same as it did in Version 1 with a minor adjustment on the FORM.OPTIONS panel.

Appendix B. QMF Objects Residing in DB2

The following tables show a DBA the QMF Version 7.2 objects that reside in the database. The tables are intended to summarize all the database objects that are needed to run QMF Version 7.2 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 93 describes the plans shipped with QMF for OS/390.

Table 93. QMF plans

Plan Name	Bind job	Notes
QMF720	DSQ1BINR	General QMF plan
DSQIN720	DSQ1BSQL	QMF plan used for installation jobs only

QMF packages

Table 94 describes the package shipped with QMF.

Table 94. QMF Packages

Package Name	Bind Job
DSQD*	DSQ1BINJ JCL (OS/390) DSQ2PREP EXEC (VM) DSQ3EDBI JCLE (VSE)
To a remote server: DSQD*	DSQ1BPKG JCL (OS/390 any supported server) DSQ2BPKB EXEC (VM any supported server) DSQ3EDBA JCLE (VSE to DB2 iSeries) DSQ3DEBU JCLE (VSE to DB2 workstation database)

QMF control tables and table spaces on OS/390

Table 95 on page 730 shows the control tables shipped with QMF. For more information about the control tables, see the *Installing and Managing QMF for OS/390* manual.

Note: iSeries requires a Collection ″Q″ be created before these QMF DB storage structures can be created. There are no nodegroups, tablespaces or dbspaces in iSeries. On VM, DBSPACEs are acquired using DSQ2DBSP EXEC.

Table 95. 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_ DIRECTORY
Q.OBJECT_DATA	DSQTSCT3	5000 primary, 200 secondary	Contains queries, forms, and procedures represented in an internal QMF format.	Q.OBJECT_ DATAX
Q.OBJECT_ REMARKS	DSQTSCT2	200 primary, 20 secondary	Contains comments that were saved when queries, forms, and procedures were created or replaced.	Q.OBJECT_ REMARKS
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 95. 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	100 primary, 20 secondary	Contains information on used by QMF during installation.. IMPORTANT: DO NOT MODIFY THIS TABLE	none

QMF control tables and table spaces on VM

Table 96. QMF Objects, Control Tables, Save Data Tables, and Sample Tables

DBspace name	QMF Table name	Index
	QMF OBJECTS	
DSQTSCT1	Q.OBJECT_DIRECTORY	Q.OBJECT_DIRECTORYX
DSQTSCT2	Q.OBJECT_REMARKS	Q.OBJECT_REMARKSX
DSQTSCT3	Q.OBJECT_DATA	Q.OBJECT_DATAX
	CONTROL TABLES	
DSQTSGOV	Q.RESOURCE_TABLE	Q.RESOURCE_INDEX
DSQTSLOG	Q.ERROR_LOG	none
DSQTSPRO	Q.PROFILES	Q.PROFILEX
DSQTSRDO	Q.DSQ_RESERVED	none
DSQTSSYN	Q.COMMAND_ SYNONYMS	Q.COMMAND_ SYNONYMNSX

QMF control tables and table spaces on VSE

Table 97. QMF Objects, Control Tables, Save Data Tables, and Sample Tables

Control table name	dbspace	dbspace 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_ DIRECTORY
Q.OBJECT_DATA	DSQTSCT3	5000 primary, 200 secondary	Contains queries, forms, and procedures represented in an internal QMF format.	Q.OBJECT_ DATAX
Q.OBJECT_REMARKS	DSQTSCT2	200 primary, 20 secondary	Contains comments that were saved when queries, forms, and procedures were created or replaced.	Q.OBJECT_ REMARKS
Q.RESOURCE_TABLE	DSQTSGOV	100 primary, 20 secondary	Contains resource control information passed to the governor exit routine.	Q.RESOURCE_ INDEX
Q.ERROR_LOG	DSQTSLOG	100 primary, 20 secondary	Contains information on system , resource, and "unexpected condition" errors. This information is more detailed than that found in error messages.	none

Table 97. QMF Objects, Control Tables, Save Data Tables, and Sample Tables (continued)

Control table name	dbspace	dbspace size (in 1K units)	Table content	Index
Q.DSQ. RESERVED	DSQTSRDO	100 primary, 20 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 98. Views shipped with QMF

View Name	Table Viewed	Operating System
Q.DSQEC_ALIASES	SYSIBM.SYSTABLES	OS/390
	SYSCAT.TABLES	workstation
	QSYS2.SYSTABLES	iSeries
Q.DSQEC_COLS_LDB2	SYSIBM.SYSCOLUMNS	OS/390
	SYSIBM.SYSTABAUTH	OS/390
	SYSCAT.COLUMNS	workstation
	SYSCAT.TABAUTH	workstation
	QSYS2.SYSCOLUMNS	iSeries
Q.DSQEC_COLS_RDB2	SYSIBM.SYSCOLUMNS	OS/390
	SYSIBM.SYSTABAUTH	OS/390
Q.DSQEC_COLS_SQL	SYSTEM.SYSCOLUMNS	VM
	SYSTEM.SYSTABAUTH	VM
Q.DSQEC_QMFOBJS	Q.OBJECT-DIRECTORY	All operating systems
	Q.OBJECT_REMARKS	All operating systems
Q.DSQEC_TABS_LDB2	SYSIBM.SYSTABAUTH	OS/390
	SYSIBM.SYSTABLES	OS/390
	SYSCAT.TABAUTH	workstation
	SYSCAT.TABLES	workstation

Table 98. Views shipped with QMF (continued)

View Name	Table Viewed	Operating System
	QSYS2.SYSTABLES	iSeries
Q.DSQEC_TABS_RDB2	SYSIBM.SYSTABAUTH	OS/390
	SYSIBM.SYSTABLES	OS/390
Q.DSQEC_TABS_SQL	SYSTEM.SYSCATALOG	VM
	SYSTEM.SYSTABAUTH	VM
Q.RESOURCE_VIEW	Q.RESOURCE_TABLE	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 VM or OS/390:

- On VM, run job DSQ2BVW EXEC.
- On MVS, run job DSQ1BVW JCL.

These jobs will DROP and CREATE all QMF control table views and GRANT necessary authorities.

On OS/390, if you want to enable QMF control table views for DB2 secondary authorization IDs, you must run one of these jobs to refresh the QMF views for that DB2 database.

VSAM clusters for OS/390

Table 99 shows the VSAM clusters shipped with QMF.

Table 99. 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
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

Table 99. VSAM clusters (continued)

Cluster Name	Object that Cluster is Needed for
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 OS/390

Table 100 describes the sample tables.

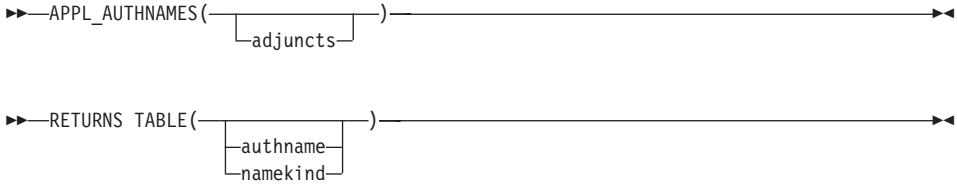
Table 100. Sample tables

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)

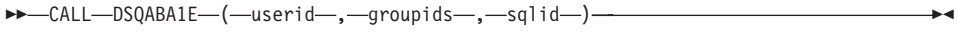
  A classification code for the name value in AUTHNAME:
  - **1** Primary authorization ID or user name
  - **2** Secondary authorization ID or group name
  - **3** Current authorization ID

    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. Migration and Fallback between QMF Releases

Note: Skip this section if you are installing QMF for the first time.

If your installation was using an earlier release of QMF before it installed Version 7 Release 2, your users might still be operating an earlier release of QMF. You need to help them operate the new release. You might need to:

- Grant them access to the QMF Version 7.2 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 described in the previous section. The migration is essentially the same when moving from the following QMF releases:

- Version 2 Release 2
- Version 2 Release 3
- Version 2 Release 4
- Version 3 Release 1
- Version 3 Release 1 Modification 1
- Version 3 Release 2

Note: QMF for VSE/ESA can only be migrated from Version 3 Release 1 Modification 1 to QMF Version 7.2.

This appendix assumes that QMF Version 7.2 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

Multiple releases of QMF can access one DB2 database, and all releases use the same QMF control tables and QMF objects.

Granting access to the QMF V7R2 application plan and packages

This procedure is the same for all the earlier releases of QMF. If the grants are still in effect, skip this section:

If during QMF installation, access to the QMF application plan (QMF720) was not granted to PUBLIC or to the user being migrated, run the following queries. (You need EXECUTE privilege on the QMF plan and packages with the GRANT option.)

```
GRANT EXECUTE ON PLAN QMF720 to authid
```

where *authid* is the authorization ID of the user being migrated. If you substitute PUBLIC for *authid*, you give everyone EXECUTE authority on the plan and the package.

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 subsystem".
- If the two releases of QMF are not on the same DB2 subsystem, read "Migrating QMF across different DB2 subsystems" on page 742.

Migrating QMF on the Same DB2 subsystem

Read this section to migrate when both releases of QMF are on the same DB2 subsystem.

Providing a QMF profile on OS/390

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 TSOID when you start QMF. Otherwise, insert a user row in Q.PROFILES with CREATOR set to the primary authorization ID.

For QMF Version 3.1.1 Users: There are no new columns in the Q.PROFILES table if you are migrating from Version 3.1.1.

For QMF Version 3.1 Users: There are no new columns in the Q.PROFILES table if you are migrating from Version 3.1.

For QMF Version 2.4 Users: The Q.PROFILES table now has one more column, ENVIRONMENT, which carries a profile parameter that applies to only the new release. This column has a single default value for its entry, as shown in Table 101 on page 741. The new column has no effect on the execution of Version 2.4; the new column is invisible to the Version 3 releases.

Important: The new column is not invisible unless QMF Version 2.4 users have installed the fix for APAR PL77029.

For QMF Version 2.2 and Version 2.3 Users: The Q.PROFILES table now has two more columns, MODEL and ENVIRONMENT, which carry profile parameters that apply to only the new release. These columns have single default values for their entries, as shown in Table 101. The two new columns have no effect on the execution of QMF Version 2.2 and Version 2.3; the new columns are invisible to these releases.

Important: The new columns are not invisible unless users have installed the fix for APAR PL77028.

MODEL and ENVIRONMENT columns added to existing rows contain NULLS. Through SET or SAVE PROFILE, users can supply only the MODEL column. You need to assign the ENVIRONMENT column.

Table 101. Columns added to the Version 2.3 profiles table

Column Name	Purpose
MODEL	Identifies each user's conceptual view of the data that Prompted Query utilizes to access the data. Default value is REL. .
ENVIRONMENT	Identifies the execution environment of the QMF session.

Migrating Q.OBJECT__DIRECTORY

Version 3.2 added three new columns to the Q.OBJECT__DIRECTORY table. If you are migrating from an older release of QMF, these three columns have no effect on the previous tables. Table 102 shows the three new columns.

Table 102. Columns added to the Version 3.2 Q.OBJECT__DIRECTORY table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/Values
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	DATE		Yes	Shows the date value for when an object was last used. The value is updated only once a day.

Migrating DPRE in ISPF from Version 2.4

If you are migrating from QMF Version 2.4 to QMF Version 7.2 and you plan to run both QMF Version 2.4 and Version 7.2 on the same database, you need to use the QMF Version 2.4 version of DPRE for both versions of QMF. To provide the QMF Version 2.4 version after installing QMF Version 7.2:

1. Rename or save the QMF Version 7.2 version of DSQABR13 for use when upgrading DPRE to Version 7 level.
2. Move the following CLISTs from QMF Version 2.4 library QMF240.DSQCLSTE to the QMF Version 7.2 library QMF720.SDSQCLTE:
   DSQABR11
   DSQABR12
   DSQABR13
3. From a QMF Version 2.4 session that has an authorization ID of Q, IMPORT and save the Version 2.4 version of QMF procedure Q.DSQAER1P. To do this, issue the QMF command:

   ```
   IMPORT PROC Q.DSQAER1P FROM 'QMF240.DSQSAMPE(DSQAER1P)'
   ```

When you are no longer using QMF Version 2.4, restore DPRE to the Version 7.2 level as follows:

1. Restore the QMF Version 7.2 version of DSQABR13 that was saved or renamed in step 1 above to QMF Version 7.2 library QMF720.SDSQCLTE
2. From a QMF Version 7.2 session that has an authorization ID of Q, IMPORT and save the Version 7.2 version of QMF procedure Q.DSQAER1P. To do this issue the QMF command:

   ```
   IMPORT PROC Q.DSQAER1P FROM 'QMF720.SDSQSAPE(DSQAER1P)'
   ```

Making objects from the earlier release available under QMF Version 7.2

If both releases of QMF are on the same DB2 subsystem, all the DB2 objects (tables and views, for example), are available under QMF Version 7.2 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 7.2. This topic is discussed in "Migrating QMF objects" on page 745.

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 7.2 users. Nor are these objects in the QMF Version 7.2 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 7.2 is in a different DB2 subsystem.

For QMF Version 3.1.1 users: There are no new columns in the Q.PROFILES table if you are migrating from Version 3.1.1.

For QMF Version 3.1 users: There are no new columns in the Q.PROFILES table if you are migrating from Version 3.1.

For QMF Version 2.4 users: The new Q.PROFILES table has an additional column, ENVIRONMENT.

For QMF version Version 2.2 and Version 2.3 users: The new Q.PROFILES table has two additional columns, MODEL and ENVIRONMENT.

For QMF Version 2.4, Version 2.3, or Version 2.2 users: The newly created table contains a single SYSTEM row. The values assigned to the columns appear in Table 103.

Table 103. 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 7.2 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.

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 491.

The PFKEYS, SYNONYMS, and RESOURCE__GROUP parameters play key roles in customizing the QMF environment. For a brief description of each, see Table 101 on page 741.

Making objects from the earlier release available under QMF Version 7.2

DB2 tables and QMF objects can be exported from a subsystem under an earlier QMF release and then imported under QMF Version 7.2.

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 OS390 Administration Guide*.
2. Load the unloaded tables into the Version 7.2 DB2 subsystem using the DB2 loader. For details on using the loader see the *DB2 UDB for OS390 Administration Guide*

.

If you have DXT installed, you can also use DXT for unloading and loading tables. DXT is an IBM licensed program product. For more information on DXT, see the *Data Extract: General Information* manual.

If the two versions of QMF are on different OS/390 systems, use the available networking facilities to send the exported objects and unloaded tables to the system containing QMF Version 7.2.

To migrate QMF queries, forms, procedures, and applications, make sure you read the following section, "Migrating QMF objects" on page 745.

If you have QMF High Performance Option (HPO) installed, you can use the QMF 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 QMF High Performance Option, see the web site at: http://www.ibm.com/software/data.

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 QMF Version 7.2. (For information about migrating back to an earlier release of QMF, see "Fallback" on page 751.)

Queries and forms

All queries and forms created under earlier releases of QMF can be used under QMF Version 7.2.

Procedures

Procedure objects that were saved or exported in Version 2.4 can be displayed or imported under Version 7.2, and they can be run if the abbreviations for commands and options (if any are used) are valid in Version 7.2. Version 2.4 procedures containing commands or applications requiring ISPF run only if QMF Version 7.2 is started as an ISPF dialog. Procedures written in English and saved or exported in QMF Version 2.4 can be imported and run without modification in a Version 7.2 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 QMF Version 7.2, 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 QMF Version 7.2 (or any QMF Version 2 or Version 3 release). For more information on installation-defined commands, read Chapter 27, "Customizing QMF Commands", on page 457.

Migrating applications

Version 2.4 applications containing commands requiring ISPF run only if Version 7.2 is started as an ISPF dialog. Applications that issue commands written in English and that run with Version 2.4 can be run without modification in a Version 7.2 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 QMF Version 7.2 and that use the callable interface, you must re-link edit your application using the QMF Version 7.2 interface module.

Form application migration aid

If your applications written for earlier versions of QMF refer to break field IDs, you might need to use the form application migration aid to use those applications with QMF Version 7.2.

With the form application migration aid, applications that contain Version 2.4 break numbers can be used with QMF Version 7.2. The form application migration aid *does not* allow you to export QMF Version 3 FORMs and use them in QMF Version 2.4 or earlier releases.

This aid, shipped with QMF Version 7.2, changes the break numbers in a version 7 form back to those used in early versions. Because the changes to the break field IDs are enough to keep applications from working properly, the object level field in the header record has been changed from 3 to 4. For more information about header records, see the *Developing QMF Applications* manual.

The form application migration aid runs when a user or application issues the EXPORT FORM command, which automatically triggers a command synonym for the aid. Because it requires REXX EXECs, the REXX interpreter must be available and QMF must be operating as an ISPF dialog.

Encourage application developers to:

1. Change applications that reference the break fields of the old numbering scheme to the new field numbers used in QMF Version 3.
2. Remove the command synonym for the migration aid from command synonym tables when all their applications are changed.

To set up the migration aid, run the following SQL INSERT query on the QMF command synonyms table, Q.COMMAND__SYNONYMS, and any other synonym tables that use the applications requiring the migration aid.

```
INSERT INTO Q.COMMAND__SYNONYMS (VERB, OBJECT, SYNONYM__DEFINITION, REMARKS)
  VALUES ('EXPORT', 'FORM', 'TSO DSQAEF0A', 'Version 7.2 Form Migration Aid')
```

Delete this entry from your command synonyms tables as soon as you have adjusted your applications to conform to the new externalized form.

Delete this entry with a query such as the following:

```
DELETE FROM Q.COMMAND_SYNONYMS
  WHERE VERB='EXPORT'
  AND OBJECT='FORM'
  AND SYNONYM_DEFINITION='TSO DSQAEF0A'
```

In the previous INSERT and DELETE queries, the keywords EXPORT, FORM, and DSQAEF0A are subject to NLS translation, so this query must be translated accordingly.

Running QMF under ISPF on OS/390

With QMF Version 7.2, review how you allocate resources for a QMF session, and how you start it. You can run QMF Version 7.2 and a previous version of QMF from the same ISPF session with ISPF LIBDEF. You cannot however, run more than one QMF session at a time or have more than one set of QMF libraries allocated at a time.

You can also use different versions of the database without ending your ISPF session by allocating the your DB2 libraries to DSQLLIB. Then use ISPF LIBDEF to make the libraries that you allocated to DSQLLIB available to ISPF.

With this version of QMF running under ISPF, QMF first looks for a program from DSQLLIB. If the program is not found in DSQLLIB or DSQLLIB is not allocated, QMF looks for a program as it has in past releases.

Other migration considerations

This section describes other migration considerations for QMF, including special considerations for the environment that is being used for QMF.

31-digit decimal support

If you are using QMF Version 3.1 (or later) and are operating in DB2 Version 2.3 (or higher), you have 31-digit decimal support. If you migrate from an earlier version of QMF, or from an earlier database version, you might want to determine which tables are impacted by 31-digit decimal support. The following query retrieves a list of the user tables that might be impacted by the DB2 tables:

```
SELECT DISTINCT(TBNAME)
   FROM SYSIBM.SYSCOLUMNS
     WHERE COLTYPE IN ('INTEGER', 'SMALLINT', 'FLOAT',
       'DECIMAL', 'DATE', 'TIME', 'TIMESTAMP')
   ORDER BY TBNAME
```

Governor

If a user wants to use the V2.4 IBM-supplied governor with QMF Version 3 and the Version 2.4 governor sets XCBPANEL to DXYEMU00 or DXYEMU01, the user must do one of the following:

- If the user wants to use the Version 2.4 governor and to run the QMF Version 3 product without ISPF, the user must remove statements that set

XCBPANEL from the governor exit. (QMF Version 3 provides a window help panel as the default instead of an ISPF panel.)

- If the user wants to use an unaltered version of the IBM-supplied governor shipped with QMF V2.4, the user should use the QMF governor shipped with QMF Version 3, and should add ISPF panels DXYEMU00 and DXYEMU01 to the QMF Version 3 ISPF panel library.

Governor in CICS

If you are using a modified or replacement version of the QMF Version 3.1 governor, you need to change the interface. The governor interface for Version 3.1.1 and later was changed to pass standard CICS parameters to the governor in the DFHCOMMA area, instead of through the first and second parameters.

If you plan to use the IBM-supplied governor, replace it with the new version.

If you have modified the IBM-supplied governor, or have re-written it, you must make a minor change to the way you access the address of the DXEGOVA and DXEXCBA control information. The governor continues to function as before, and its contents are unchanged.

User edit routine in CICS

The user edit routine, as documented in QMF Version 3.1 APAR PN07713, is part of the base QMF Version 3.1.1 and later products. If you created your user edit exit module DSQUECIC following the instructions in PN07713, it can be used without change in QMF Version 3.1.1 or later.

User edit routine in TSO and native OS/390 batch

For QMF Version 7.2, you need to relink your user edit code. For more information about relinking your user edit code, see Chapter 29, “Creating Your Own Edit Codes for QMF Forms”, on page 497.

Callable interface in CICS

If you are migrating from QMF Version 3.1 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.

Printing in CICS

In QMF Version 3.1, when no printer is specified on the PRINT command, output is held in CICS auxiliary temporary storage, replacing the previous report. In Version 3.2 and later, the report is not replaced. If CICS auxiliary temporary storage already exists, QMF appends the report to the existing temporary storage queue.

Export/import support for CICS on OS/390

QMF V3.1.1 support for export/import in the CICS/MVS environment uses TSO file support. This level of support is not recommended when running in a CICS environment. In fact, some error conditions can cause the entire CICS region to abnormally terminate. In QMF Version 3.2 and later, this problem is corrected; TSO file system support is replaced by support for CICS temporary or transient data.

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 7.2 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 2

QMF Version 3 contains a new DD statement and dataset, DSQPNLE, and ISPTLIB is no longer used.

The message tool is no longer shipped with the product. To see a message help panel, issue the following command on the QMF command line:

```
HELP msgno
```

For more information on messages, see the *QMF Messages and Codes* manual .

The QMF demonstration application for function key customization is no longer shipped with QMF.

The QMF ISPF panel library data set, SDSQPLBE (formerly DSQPLIBE), has decreased in size of members from about 2500 members to about 70 members due to QMF's conversion from ISPF panels to GDDM mapped panels.

The following views, available in QMF Version 2, do not exist in Version 3:
- Q.AUTH__LIST
- Q.COLUMN__LIST
- Q.TABLE__LIST
- Q.QUERY__LIST
- Q.PROC__LIST
- Q.FORM__LIST
- Q.QMFTABLE__LIST

For information about how to obtain these object lists in Version 3, see "Customizing a user's database object list" on page 366.

Global variables and the governor on VM

These are the changes made to the global variables and the governor exit control block for QMF Version 7.2:

- Global variables

  DSQA0_QMF_RELEASE is now set to '13'

  DSQA0_QMF_VER_RELS is now set to 'QMF V7R2.0'
- Governor exit control block DXEXCBA fields

  XCBRELN is now set to '13'

  XCBQMF is now set to 'QMF V7R2.0'

Use of the invocation procedure

In the current release, if you start QMF with a value for the DSQSRUN program parameter, the specified invocation procedure is run at the initial location (the first server). If you change locations during the QMF session, the invocation procedure is not rerun. Instead, the following QMF session, the invocation procedure is not rerun. Instead, the following message is displayed on the Home Panel:

```
"Your invocation procedure was not rerun due to location differences "
```

This remains until:
- You connect to the initial location.
- You set the global variable DSQEC_RERUN_IPROC to 0.
- You issue the EXIT command.

Q.VPROFILE

In previous releases, Q.VPROFILE was created when you installed QMF into a database. This view is no longer created during QMF installation. If you are |migrating to QMF Version 7.2 from an existing release of QMF that is already installed into that database, the Q.VPROFILE view remains intact; that is, QMF does not drop it during the migration process. If you have any applications that depend on Q.VPROFILE, and you are installing QMF Version 7.2 into a new database, you must create Q.VPROFILE or use your own view in the application. If you need to use Q.VPROFILE, use the following statement to create it:

```
CREATE VIEW Q.VPROFILE AS SELECT *FROM Q.PROFILES WHERE CREATOR =
'SYSTEM'
```

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 OS/390. Cleanup is described in "Step 36—Clean up after install" on page 92 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 7.2 to an earlier release. If you are interested only in fallback, turn to "Fallback". Cleanup is described in "Step 36—Clean up after install" on page 92 and is not discussed here.

Version 3 (and later) forms and the forms for earlier QMF versions have different internal representations. When one of the earlier forms is converted to a Version 3 (or later) form, it no longer can be used with an earlier version of QMF.

Applications developed to work with QMF Version 2.4 (or earlier) forms might also not work if they reference the object level in the header record or break field numbers.

To prevent users from accidentally converting a version 2 or earlier form to a Version 7.2 form (a mistake a user might make by replacing the old form while running under QMF Version 7.2), you can save both the earlier version and a Version 7.2 version with different names until it is clear that one version is no longer needed. For example, the following commands convert work area forms to Version 7.2 forms:

```
SAVE FORM AS FORM1
EXPORT FORM TO FORM2
```

If your installation falls back to the earlier QMF release, urge your users to recreate their Version 7.2 objects under the earlier release before QMF Version 7.2 is withdrawn.

Note: REXX is not supported in CICS.

Re-establishing the earlier profiles

The ENVIRONMENT column, absent in QMF Version 2.4, does not affect the Version 7.2 profile. The same holds true for all the columns added since Version 2.2.

Note: 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 QMF Version 7.2 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 745.

If you fall back to the earlier QMF release, some objects created under QMF Version 7.2 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 7.2 objects in earlier releases.

- Forms

  Form objects that are saved or exported from Version 7.2, and displayed or imported to earlier releases of QMF, can be expected to execute normally. However, form objects saved or exported from Version 7.2 cannot be used in Version 2.4 or earlier.

  Before they can be used in earlier applications, forms exported from Version 7.2 that use break field numbers (or the object level in the header record) require the Form Application Migration Aid, as described in "Form application migration aid" on page 746.

- Queries

  Some restrictions apply to Version 7.2 queries for fallback to earlier releases:

  – SQL queries: You can export SQL queries from Version and import them on earlier releases, and they execute normally. However, SQL queries saved on Version 7.2 cannot be used in Version 2.4 or earlier.

  – Prompted queries: You can display and import Version 7.2 prompted queries in earlier releases provided they do not contain variables, or expressions with more than the old 55 or 65 character limit.

  - QBE queries: Queries created with QBE (Query-by-Example) saved or exported in Version 7.2 can be displayed or imported in earlier releases and execute normally.
- Procedures

  Procedure objects exported from Version 7.2 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 7.2 cannot be displayed with earlier releases unless you first export them from Version 7.2 and import them into the earlier release. Procedures with logic, that is, procedures that contain REXX logic, cannot be displayed or imported in releases earlier than Version 3.
- 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 7.2 commands under earlier releases" for details.
- Applications that call the callable interface

  Applications call the callable interface in their CLISTs and programs to call QMF. The callable interface was introduced for Version 2.4, so applications running with earlier versions of QMF cannot use it.

For specific differences between an earlier QMF release and QMF Version 7.2, compare the two releases of the *QMF Reference* manual.

Using QMF Version 7.2 commands under earlier releases

Version 7.2 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 E. How QMF and GDDM Programs are Defined to CICS

QMF for OS/390 and QMF for VSE/ESA provide 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/MVS and CICS/VSE

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 xiv. The transaction ID is defined in either the CICS program control table (PCT) or the system definition (CSD) file. If you need to, you can change this default transaction ID:

- To update the CSD, see the *CICS/MVS Resource Definition (Online)*
- To update the PCT, see the *CICS/MVS Resource Definition (Macro)*
- To update the CSD, see the *CICS/VSE Resource Definition (Online)*
- To update the PCT, see the *CICS/VSE Resource Definition (Macro)*

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 758 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).

For more information on the performance implications of nonresident programs, see the *CICS/MVS Performance Guide* or the *CICS/VSE Performance Guide* .

Loading QMF to the 31-Bit shared virtual area on VSE

The default installation loads QMF to an individual CICS partition. Depending on the configuration of your system, you might consider loading QMF programs to the 31-bit VSE shared virtual area (SVA).

If several CICS partitions run QMF, consider loading QMF to the SVA rather than to the resident area in the individual CICS partitions where QMF is running. Loading QMF to the SVA:

- Allows two or more CICS systems in the same processor to share QMF programs. The CICS systems do not have to be using intercommunication facilities to benefit from sharing programs.
- Automatically protects the programs from being overwritten by other programs, such as CICS applications. This integrity also applies to a single CICS system within the processor.

If you decide to load programs to the SVA, you need to use the SVA command to allocate space in the SVA for QMF modules and their system directory list (SDL) entries at IPL time. The space you allocate is in addition to any required for other phases in the SVA.

The following QMF base programs can be loaded to the SVA; they take approximately 2.8 MB of space:

DSQQMF
: Main QMF program

DSQCBST
: Driver for the callable interface

DSQQMFE
: Identifies the environment and language being started

DSQCELTT
: Holds messages and constants for QMF objects and screen displays

DSQCEBLT
: Holds command definitions and permits bilingual support

DSQUEGV3
: Required for the governor exit routine (discussed in Chapter 14)

DSQUECIC
: Required for user-written edit exit routines (discussed in Chapter 13)

DSQUOPTS
: Required for user-written edit exit routines

If you are using an NLF: You can also load the following QMF NLF programs to the SVA. These programs require a total of approximately 300 KB in the SVA, per NLF. The *n* symbol represents an NLID from table xx on page yy.

- DSQQMF*n*
- DSQC*n*LTT
- DSQC*n*BLT
- DSQU*n*GV3

To load programs into the SVA, issue a SE SDL command from the background (BG) partition, naming the selected programs. You can issue this command at any time after IPL, but you must issue it before bringing up any CICS system that uses programs from the SVA.

VSE loads SVA phases from the libraries of the BG partition library search chain. You need to specify the QMF library in the SEARCH operand of the LIBDEF statement for the BG partition. shows an example of a load list for QMF programs:

```
//  JOB CICS SVALOAD
*   CICS/VSE/EDA SVA LOAD LIST
*   LIBDEF statement for the QMF sublibrary
SET SDL
DSQQMFE,SVA
DSQCBST,SVA
DSQCELTT,SVA
   . . .
   . . .
   . . .
/*
/&
```

Figure 275. Loading QMF programs to the SVA

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 the:

GDDM Installation and System Management for OS/390 or VSE manual for GDDM, and

GDDM System Customization and Administration manual for GDDM.

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.

If you use GDDM 2.3, you need GDDM-PGF 2.1.1.

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*.

See “Program parameters for OS/390 and z/OS” on page 259 for information on how QMF uses temporary storage in the CICS region on OS/390 or how QMF uses GETVIS storage in the CICS partition on VSE.

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, 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.

Any references in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is as your own risk.

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
J74/G4
555 Bailey Avenue
San Jose, CA 95161-9023
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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AIX
AIX/6000
C/370
CICS
CICS/ESA
CICS/MVS
CICS/VSE
COBOL/370
DATABASE2
DataJoiner
DB2
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
DXT
GDDM
IBM
IBMLink
IMS
iSeries
Language Environment
MVS
MVS/ESA
MVS/XA
OfficeVision/VM
OS/2
OS/390
PL/I
PROFS
QMF
RACF
S/390
SQL/DS
Virtual Machine/Enterprise Systems Architecture
Visual Basic
VM/XA
VM/ESA
VSE/ESA
VTAM
z/OS

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other counrtries, or both.

Lotus and 1-2-3 are trademarks of Lotus Development Corporation in the Unites States, other counrties, or both.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), 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 CMS or TSO command that invokes an EXEC or CLIST, respectively.

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.

For a list of QMF publications, see "The QMF library" on page ix.

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 Server for OS/390 and z/OS publications

DB2 Universal Database for OS/390 and 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

Document Composition Facility (DCF) publications

DCF and DLF General Information

Distributed Relational Database Architecture (DRDA) publications

Every Manager's Guide
Connectivity Guide

DXT publications

DXT Guide to Dialogs
Data Extract: Planning and Administration Guide for Dialogs
Data Extract: User AE's Guide
Learning to Use DXT

Graphical Data Display Manager (GDDM) publications

GDDM General Information
GDDM Base Application Programming Reference
GDDM User's Guide
GDDM/VSE Program Directory
GDDM Messages

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

Bibliography

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

C

D

E

F

G

H

I

J

K

L

M

N

Q

R

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

Query Management Facility™
Installing and Managing QMF
Version 7 Release 2

Publication No. GC27-0720-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
GC27-0720-02

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
Department HHX/H3
555 Bailey Ave.
San Jose, CA
U.S.A.
95161-9023

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Program Number: 5675-DB2

Printed in U.S.A.

GC27-0720-02